

AFIT/GOR/ENS/95M-18



PROACTIVE MONITORING OF PERFORMANCE  
IN STOCHASTIC COMMUNICATION NETWORKS

THESIS

John C. Van Hove, Captain, USAF

AFIT/GOR/ENS/95M-18

Approved for public release; distribution unlimited

19950503 123

# PROACTIVE MONITORING OF PERFORMANCE IN STOCHASTIC COMMUNICATION NETWORKS

## THESIS

Presented to the Faculty of the School of Engineering  
of the Air Force Institute of Technology  
Air University

In Partial Fulfillment of the  
Requirements for the Degree of  
Master of Science in Operations Research

John C. Van Hove, B. S.  
Captain, USAF

March, 1994

Accession For	
NTIS	CRA&I <input checked="" type="checkbox"/>
DTIC	TAB <input type="checkbox"/>
Unannounced <input type="checkbox"/>	
Justification _____	
By _____	
Distribution /	
Availability Codes	
Dist	Avail and/or Special
A-1	

Approved for public release; distribution unlimited

## THESIS APPROVAL

STUDENT: John C. Van Hove, Captain, USAF

CLASS: GOR-95M

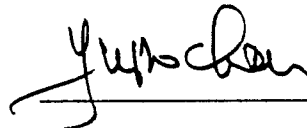
THESIS TITLE: Proactive Monitoring of Performance in Stochastic Communication Networks

DEFENSE DATE: 23 February 1995

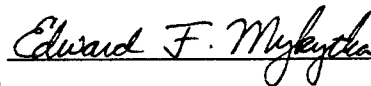
Committee: Name/Title/Department

Signature

Advisor YUPO CHAN  
Professor of Operations Research  
Department of Operational Sciences

A handwritten signature in black ink, appearing to read 'Yupo Chan', written over a horizontal line.

Reader EDWARD F. MYKYTKA  
Associate Professor of Operations Research  
Department of Operational Sciences

A handwritten signature in black ink, appearing to read 'Edward F. Mykytka', written over a horizontal line.

## *Preface*

This research investigated methods for modelling dynamic network performance in computer communication networks. The methodologies developed were the result of six months of research into stochastic network throughput and reliability, dynamic network flows, and vector time-series analysis. I would like to thank my thesis advisor, Dr. Yupo Chan, for his technical expertise and guidance. I would also like to thank my reader, Dr Edward Mykytka, for his insights and suggestions. Finally, I thank my wife, Tammy, for her patience and understanding over the last year and a half.

John C. Van Hove

## *Table of Contents*

	Page
I. Introduction .....	1
1.1 Background .....	1
1.2 Research Problem .....	4
1.3 Research Objectives .....	4
1.4 Assumptions .....	5
1.5 Scope of Research .....	5
II. Literature Review .....	8
2.1 Network Representation .....	8
2.1.1 Network Components .....	8
2.1.2 Failing Components .....	9
2.2 Network Performance .....	10
2.2.1 Network Performance Measures .....	10
2.2.2 Network Throughput .....	11
2.2.3 Network Reliability .....	15
2.3 Dynamic Networks .....	18
2.3.1 Stochastic Programming .....	18
2.3.2 Dynamic Network Flow .....	19
2.4 Summarizing Vector Time-Series .....	21
2.4.1 MARMA Models .....	21
2.4.2 Simultaneous Estimating Equations .....	22

	Page
III. Methodology .....	23
3.1 Understanding the Problem .....	23
3.2 Modelling Absolute Performance Measures .....	27
3.2.1 Absolute Performance .....	27
3.2.2 Network Throughput .....	27
3.2.3 Network Reliability .....	29
3.3 Modelling Relative Performance Measures .....	31
3.3.1 Relative Performance .....	31
3.3.2 Bit Error Rate (BER) .....	31
3.3.3 Delay .....	33
3.4 Path Enumeration Algorithm .....	34
IV. Implementation .....	37
4.1 Software Selection Decisions .....	37
4.2 Program Design .....	40
4.2.1 Input .....	40
4.2.2 Path Enumeration .....	41
4.2.3 Model Construction .....	41
4.2.4 State Vector Generation .....	42
4.2.5 Model Solution .....	43
4.3 Program Control .....	43
4.4 Summary .....	44

	Page
V. Results and Analysis .....	46
5.1 Analyzing a Sample Network .....	46
5.1.1 Network Description .....	46
5.1.2 Bounding Network Performance .....	49
5.1.3 Estimating Performance Data .....	51
5.2.1 Network A .....	54
5.2.2 Network B .....	60
5.2.3 Network C .....	62
VI. Validation .....	66
6.1 Obtaining Data Sets .....	66
6.2 Calibrating Estimating Equations .....	68
6.3 Analyzing Time-Series Residuals .....	70
6.4 Overall Model Validity .....	72
VII. Conclusions and Recommendations .....	74
7.1 Summary .....	74
7.2 Conclusions .....	75
6.3 Recommendations .....	77
Bibliography .....	78

	Page
Appendix A. Fortran Implementation .....	81
A.1 Sample Input .....	81
A.2 Input Subroutines .....	82
A.3 Path Enumeration Subroutines .....	83
A.4 Model Construction Subroutines .....	87
A.5 State Vector Generation Subroutines .....	89
A.6 Model Solution Subroutines .....	91
A.7 Main Program and Makefile .....	101
 Appendix B. Results For the Sample Network .....	 108
B.1 Sample Network Description File .....	108
B.2 Sample Network Path Enumeration .....	109
B.3 Sample Network Program Output .....	110
 Appendix C. Results For Network A .....	 115
C.1 Network Description File For the Original Network A .....	115
C.2 Path Enumeration For the Original Network A .....	116
C.3 Program Output For the Original Network A .....	120
C.4 Network Description File For the New Network A .....	123
C.5 Path Enumeration For the New Network A .....	124
C.6 Program Output For the New Network A .....	128



	Page
Appendix D. Results For Network B .....	131
D.1 Network Description File For Network B .....	131
D.2 Path Enumeration For Network B .....	133
D.3 Program Output For Network B .....	143
Appendix E. Results For Network C .....	146
E.1 Network Description File For Network C .....	146
E.2 Path Enumeration For Network C .....	148
E.3 Program Output For Network C .....	158
Appendix F. Validation Data and Output .....	161
F.1 Original Estimated Data Set .....	161
F.2 Observed Data and Aggregate Estimated Data .....	166
F.3 Output From Estimating Equation Calibration .....	167
F.4 Output From Time-Series Residual Analysis .....	171

## *List of Figures*

	Page
Figure 1: Representing a Network as a Tree Structure .....	35
Figure 2: FORTRAN Implementation Components .....	39
Figure 3: Sample Network .....	47
Figure 4: Time-Expanded Replica of the Sample Network .....	48
Figure 5: Reduced Time-Expanded Replica of the Sample Network .....	49
Figure 6: Estimated Vector Time-Series Performance Data .....	52
Figure 7: Topology of Network A .....	55
Figure 8: Estimating Static Performance .....	58
Figure 9: Topology of Network B .....	60
Figure 10: Topology of Network C .....	63
Figure 11: Validation Data Sets .....	68
Figure 12: Time-Series Residual Plots .....	71

## *List of Tables*

	Page
Table 1: Node-Arc LP Fomulation of Max Flow Problem .....	12
Table 2: Arc-Path LP Fomulation of Max Flow Problem .....	12
Table 3: Exact Expected Max Flow Model .....	13
Table 4: Arc-Path LP Fomulation of the Lower Bound Model .....	14
Table 5: Arc-Path LP Fomulation of the Upper Bound Model .....	15
Table 6: Abraham's Boolean Algebra Algorithm for Disjoint Products .....	16
Table 7: Maximum Dynamic Flow Model Using Time-Expanded Network .....	20
Table 8: More Efficient Maximum Dynamic Flow Model .....	21
Table 9: Max Dynamic Flow Lower Bound Model Fomulation .....	27
Table 10: Max Dynamic Flow Upper Bound Model Fomulation .....	28
Table 11: Bounding Reliability With the Most Probable States .....	30
Table 12: Upper and Lower Bound Models For Min Cost Flow .....	32
Table 13: Bounding BER From Min Cost Flow Models .....	33
Table 14: Bounding the Average Delay Performance Measure .....	34
Table 15: Sample Network Expected Performance Bounds .....	50
Table 16: Node Parameters For Network A .....	55
Table 17: Arc Parameters For Network A .....	56
Table 18: Expected Performance Bounds For Network A .....	59
Table 19: Arc Parameters For Network B .....	61
Table 20: Expected Performance Bounds For Network B .....	62
Table 21: Arc Parameters For Network C .....	64

	Page
Table 22: Expected Performance Bounds For Network C .....	65
Table 23: Calibrated Estimating Equations .....	69

## *Abstract*

This research proposes several models for placing bounds on the expected values of some dynamic performance measures for computer communication networks with failing components. These models provide an understanding of expected network performance that is useful in the process of proactive performance monitoring and also in defining level of service agreements with network users. There were three objectives for this research. The first objective was to extend some existing models of steady-state stochastic network performance to a dynamic network flow representation in order to capture the dynamic nature of proactive monitoring. The second objective was to convert the extended absolute performance models to relative performance models that are dependent on the utilization level of the network. This was accomplished by converting a maximum network flow model for throughput to a minimum cost flow model with a constant level of source to sink network flow. The final objective was to demonstrate a methodology for validating the proposed models against an operational communication network. This was accomplished by collecting actual vector time-series performance data, using the models to estimate a similar data set, and performing some multivariate analysis with the two data sets.

# PROACTIVE MONITORING OF PERFORMANCE IN STOCHASTIC COMMUNICATION NETWORKS

## ***I. Introduction***

This chapter begins with a brief background on the growing use of computer communication networks and the resulting need for proactive monitoring of network performance. Next, the research problem is defined and stated in terms of specific research objectives. Then, assumptions are listed that will be used throughout the entire research effort. Finally, there will be a discussion on the scope of the research outlining the problem areas that will be considered in this effort.

### ***1.1 Background***

A growing number of organizations within the United States military are becoming increasingly dependent on the capability to efficiently transmit vast quantities of information between remote sites by way of computer communication networks. Advances in the capability and accessibility of computer networks are bringing some organizations to rely on these networks for routine daily operations. Computer communication networks are made up of imperfect components and so are subject to component failures leading to degraded network

capability. With the growing reliance on computer network assets, comes a need for some means to proactively monitor network performance to identify and avoid potential problems associated with degraded network capability.

Performance measures must be identified for use in the proactive monitoring of performance in a communications network. These performance measures need to provide insight into the overall performance trend of the network so that critical network degradation can be avoided. Three performance measures that have been identified as candidates for this study are network reliability, availability, and degradation. By monitoring these measures over time and evaluating the multivariate and temporal correlation in the resulting vector time-series data, it is possible to gain insight regarding the relationships between these measures. An understanding of how a set of performance measures describe the state of the system could be useful in identifying the potential for critical network degradation.

In reference to a computer communication network, the network performance measures reliability, availability, and degradation have exact definitions specific to the field of digital transmission systems. Reliability is used to describe the probability that a system will perform without failure for a given period of time. For systems of electronic components, this is normally represented by a random variable having an exponential distribution [29:40]. Availability is a term for either the probability that a system will be functional at a given point in time or the proportion of time that the system will be functional [29:41].

Degradation in a computer communication network can be stated in terms of error performance which is traditionally expressed by four main measures. Bit error rate (BER) is the ratio of errored bits to the total number of bits transmitted. Error-free seconds (EFS), or error seconds (ES), are the percentage or probability of one-second intervals free of error for

EFS or in error for ES. Severely errored seconds (SES) or degraded minutes (DM) are used to show the percentage of time that the BER exceeds a set threshold value for a given interval of time. Error-free blocks (EFB) are the percentage or probability of data blocks that do not include any errored bits. All four of these measures attempt to quantify degradation in terms of the same network characteristic, error bits [29:48-49].

A separate set of measures are used to gauge performance in causal models for stochastic networks and these performance measures have unique definitions that are not necessarily the same as those for digital transmission systems. The performance measures throughput and reliability for a causal network model cover the same aspects of performance as reliability, availability, and degradation in a computer communication network. In a causal network model, reliability is a measure of the probability of connectivity between source and sink. This is very similar to availability in a computer communication network and it is based on the 'reliabilities' of the electronic components making up the system. Throughput in a causal network model is a measure of the maximum flow between source and sink per unit time. Causal model throughput and communication network degradation are similar in that the latter measures the percentage of error bits while the former measures percentage of non-error bits. No direct functional relationship is implied, but there is a strong inverse tendency such that more degradation implies less throughput.

A computer communication network can be represented with a stochastic network model and the causal network performance measures throughput and reliability can be studied as surrogates to reliability, availability, and degradation in the communication network. The stochastic network model is composed of a set of nodes connected by a series of arcs, where all network components, arcs and nodes, are subject to periodic failure. The nodes in this



model represent remotely located users and switching centers of the communication network while the arcs stand for the communication lines connecting these remote sites. Associated with every arc and node is a reliability parameter governing the component's failure rate which plays a key role in the performance of the system.

### ***1.2 Research Problem***

The ultimate objective of this research effort is to construct models for the expected value of the type of performance measures that are operationally monitored over time in a computer communication network. The approach is to use dynamic network flow models with a stochastic network representation of a communication network. These performance measure models are used to generate bounds for the expected values of the vector time-series performance data that would be likely output of network monitoring. The idea is to use the causal models to understand what the expected network performance for a given network is. Such an understanding can be applied to recognize the potential for critical network degradation. The models are also useful in evaluating the effect of change to the network structure resulting from failure or modification.

### ***1.3 Research Objectives***

The following specific research objectives must be accomplished in order to complete this research effort:

1. Demonstrate a computer communication network can be represented with a stochastic network model.
2. Identify the most appropriate models for stochastic network throughput and reliability.

3. Modify the performance models to a dynamic network flow representation.
4. Construct formulations for digital transmission system performance measures that are relative to utilization level and candidates for operational monitoring.
5. Apply the proposed dynamic network performance models to realistic computer communication network topologies and validate with actual data.

#### ***1.4 Assumptions***

The following set of simplifying assumptions are necessary to restrict the problem area and are used throughout the entire research effort:

1. All network components that are subject to periodic failures can be modelled with a binary variable indicating that the component is either functional or non functional (degraded component functionality will not be considered).
2. Routing of flow between a source and a sink in the network model is accomplished based on a shortest path algorithm.
3. Single-commodity, multiple-path networks will be used to construct the performance measure models.
4. The network performs at its best. This means that if a component is available and needed it will be used (no prioritized access to resources).

#### ***1.5 Scope of Research***

The first part of this research, representing a computer communication network with a stochastic network model and constructing models for network performance, has been addressed in previous thesis efforts (Yim [31], Jansen [17], and others). The important feature

of this stage of the research effort is defining performance measure models that are computationally feasible for large networks and which can be easily extended to a dynamic network flow model. The dynamic network flow representation is necessary to model the dynamic nature of the performance measures that would likely be periodically sampled from an operational computer communication network. In a standard stochastic network model, the throughput measure represents the maximum source to sink flow *per unit time* while in the dynamic model it is the maximum number of flow units that can go from source to sink *per time interval*. Temporal correlation is insured by computing dynamic network performance measures for sequential, overlapping time intervals.

The performance measure models, once extended to dynamic network flow, are used as the basis for formulating new measures that are more likely to be monitored on an operational computer communication network. These performance measures will be relative to the utilization level of the network rather than absolute limits for a given network. The findings here are intended to be useful in understanding expected network performance in terms of a set of relative performance measures and studying how sensitive these expectations are to network alterations. This effort will identify and model network performance measures that are promising candidates for proactive monitoring. A parallel research effort is investigating methods of tracking these measures over time and identifying when network performance is out of bounds. The understanding of expected network performance gained from the causal models developed here may be useful to the development of methodologies in the parallel effort.

The final stage of this research is gauging the appropriateness or validity of the suggested models. Vector time-series performance data produced with the causal models is

evaluated against actual observed communication network performance data. The observed data sample must be in the same form as the data produced by the causal model constructed here. It should be vector time-series data for network performance in terms of a common set of network performance measures. The ideal source for this data sample is an operational computer communication network conforming to the assumptions stated previously.

## ***II. Literature Review***

This chapter reviews literature applicable to proactively monitoring performance in stochastic communication networks. First, there is a discussion of how a communication network can be represented by a stochastic network model. Then, a development of some stochastic network performance measures is presented along with models for upper and lower bounds on these measures. Next, a dynamic network flow model is presented as one possible technique to add a temporal dimension to network performance. Finally, there is a discussion of possible strategies for summarizing the characteristics of vector time-series performance data for the purpose of model validation.

### ***2.1 Network Representation***

**2.1.1 Network Components.** To model a communication network it is necessary to represent all of the network components as stochastic arcs and nodes. In this case, all users and switching centers become the nodes in the network while the communication lines connecting these nodes serve as the network arcs. Information flows through the network in accordance with known parameters of the arcs and nodes. Among the nodes are designated origin-destination pairs that have an associated parameter representing the amount of flow required to be transmitted between them. The remaining nodes are transitory being neither a source nor a sink for network flow. Three parameters are necessary to describe each network arc. One arc parameter defines the maximum flow of information across the arc per unit time, one is the traversal time for the arc, and the last is the reliability of the arc. A collection of arcs that forms a chain between a source node and a sink node in a network is called a path.

Two common constraint sets for network flow models deal with conservation of flow and arc capacity. Conservation of flow constraints in a network model simply restrict the sum of all flow into a node to be equal to the sum of all flow out of that node. In general, this equality must hold for every node represented in the network. Arc capacity constraints set limits on the maximum, or minimum, amount of flow per unit time across each of the arcs in the network. It is often useful to summarize the relationships between the components of a network in either a node-arc or arc-path incidence matrix. These matrix representations can be useful in defining a compact formulation of the constraints in linear model formulations.

The node-arc incidence matrix has a row for every network node and a column for each network arc. The element in row  $i$ , column  $j$ , of this matrix has a value of one if arc  $j$  originates at node  $i$ , a value of negative one if arc  $j$  terminates at node  $i$ , or a value of zero if arc  $j$  is not connected to node  $i$ . The arc-path incidence matrix has a row for every arc in the network and a column for each distinct path between the origin and destination nodes. The element in row  $i$ , column  $j$  of this matrix has a value of one if arc  $i$  is a member of path  $j$  between the origin-destination pair. When arc  $i$  is not a member of path  $j$ , the matrix element has a value of zero.

**2.1.2 Failing Components.** One way to model the stochastic nature of the components in these networks is to assign a binomial failure rate to every component in the system. Such a rate would represent the probability at any point in time that the associated component is functional. This definition of component status assumes that components can either be completely functional or non-functional with no allowance for degraded component capability. Networks conforming to this generalization of the stochastic network model are classified as stochastic binary systems (SBS) [6:154]. These networks may be further

simplified by assuming that all nodes are 100% reliable so that only arcs may fail. Failing nodes may be represented as two 100% reliable nodes connected by an arc with the failure rate of the original node [31:10].

In this formulation it is assumed that each network component fails independent of the rest of the system. This feature allows path reliabilities to be obtained as a function of the component arc reliabilities. Since a path is composed of a set of arcs connected in series, the reliability of a path is simply the product of all of the component arc reliabilities. It should be noted here that although it may be feasible to model network arcs with independent failure rates, the same is not true for network paths. Two distinct paths between a given origin and destination may contain one or more common arcs. If paths share common component arcs, then the path reliabilities are not independent.

## ***2.2 Network Performance***

***2.2.1 Network Performance Measures.*** Performance measures for networks containing failing components have been developed to optimize network design, compare performance between existing network designs, or just to analyze performance of a single network. Examples of performance measures used for these purposes include throughput (maximum flow), reliability, delay, and several measures of errored bits. The sponsor for this research effort originally identified reliability, availability, and degradation as potential candidates for monitoring. In the field of digital transmission systems, the term reliability is a measure of the probability that a component or system will function without failure for a given period of time (normally exponentially distributed). Typically, in the analysis of network performance, reliability is more frequently used to refer to the proportion of time a component

or system is expected to be functional. This is the same definition given to a measure of performance known as availability in digital transmission systems [18:308]. Throughout this document, discussion of monitoring network reliability shall refer to the measure of digital transmission system performance known as availability.

When evaluating the performance of a network, it is important not to focus on any single measure of performance. Aggarwal suggests that this leads to erroneous conclusions about the performance of the network [2:184]. Kubat [18] makes the following argument for the joint assessment of system performance (throughput) and reliability:

Perhaps the most important factors in judging the quality of a communication or computer network are its performance and reliability. If two systems have the same performance characteristics, one anticipates that in the long run more reliable systems will perform better than less reliable ones. However, when comparing two (or more) systems with both different performance characteristics and different reliabilities (availabilities), it is not always clear which system will be superior overall.

Kubat is talking about using performance measures to compare multiple designs or existing networks, but the same ideas apply to monitoring performance in a single network. The approach is to identify an appropriate set of performance measures for monitoring so that a complete picture of network performance is being considered.

**2.2.2 Network Throughput.** One measure of network performance, throughput, focuses on gauging the maximum source to sink capacity of a given network. Deterministic throughput is obtained by maximizing total source to sink flow subject to conservation of flow and arc capacity constraints. The model for deterministic throughput can be formulated as a simple linear program using either a node-arc or arc-path constraint representation. Table 1 shows the node-arc linear programming formulation of the max flow problem and Table 2 shows the arc-path formulation.



```

Model Parameters:
  uj  -- the capacity of arc 'j'
  A    -- the node-arc incidence matrix
  bi  -- -V if 'i' is source node
        -- 0 if 'i' is intermediate node
        -- V if 'i' is sink node

Decision Variables:
  xj  -- the flow across arc 'j'
  V    -- the total network flow

Model Definition:
  MAX V    -- maximize total network flow
  ST Ax=b  -- conservation of flow
      0 ≤ xi ≤ ui  -- arc flow capacities

```

**Table 1: Node-Arc LP Formulation of Max Flow Problem**

In the case of the node-arc model, a variable is included to represent total network flow leaving the source node or entering the sink node. This single variable is maximized subject to the network constraints summarized using the node-arc incidence matrix. For the arc-path formulation, there is no single variable that keeps track of the total source to sink flow. Instead, the sum of all path flows is maximized achieving the same objective.

Calculating deterministic network throughput, either by brute force simplex or by network algorithms that capitalize on unique characteristics in network models, is not computationally tasking even for large networks. When failing components are introduced

```

Model Parameters:
  ui  -- the capacity of arc 'i'
  A    -- the arc-path incidence matrix

Decision Variables:
  fj  -- the flow across path 'j'

Model Definition:
  MAX ∑ fj  -- maximize the sum of path flows
  ST Af ≤ u  -- arc flow capacities
      fj ≥ 0  -- only positive path flows

```

**Table 2: Arc-Path LP Formulation of Max Flow Problem**

into the model, though, computational feasibility does become an issue [5,10]. To obtain an exact value for the expected max flow in a network with failing components, it is necessary to solve the deterministic model once for every possible state that the network could be in. The state of the network is defined by the status, up or down, of all the network components. The expected max flow is the sum of all state deterministic solutions weighted by the associated state probabilities as shown in Table 3. The number of possible states for a network is given by  $2^n$  where  $n$  is the number of failing components. It is easy to see that the number of states grows exponentially with the number of failing components and complete enumeration quickly becomes an infeasible solution alternative.

Model Parameters:	
$P_i$	-- the probability of being in state 'i'
$V_i$	-- the deterministic max flow for state 'i'
Model Definition: $\text{MAX } \sum P_i V_i$	

**Table 3: Exact Expected Max Flow Model**

To get around the computational problems associated with throughput computation in large networks it is necessary to use solution algorithms designed specifically for network type problems and focus on estimates rather than exact values. Sancho [27] presents an algorithm that uses dynamic programming for path selection to find a network's expected max flow. It is also possible to estimate the expected max flow by considering a most probable subset of the set of possible paths. This approach builds on the fact that a small portion of the set of all network states probabalistically dominates the rest of the set. This idea may be applied to obtain an estimate of the expected max flow by evaluating a small portion of the possible network states instead of total state enumeration.

Another way to handle the issue of computational feasibility is to develop models for bounds on the exact expected max flow [10,22]. Yim [31] presents a concise development of models that do just that, bound the expected max flow in networks with failing components. These models build on the standard arc-path formulation of the max flow problem presented previously by introducing the component reliabilities. Table 4 shows the first of these, a model for the lower bound on the expected maximal flow.

```

Model Parameters:
    ui -- the capacity of arc 'i'
    A -- the arc-path incidence matrix
    Rj -- the reliability of path 'j'

Decision Variables:
    fj -- the flow across path 'j'

Model Definition:
    MAX  $\sum R_j f_j$  -- maximize the sum of path flows
    ST   $Af \leq u$  -- arc flow capacities
         $f_j \geq 0$  -- only positive path flows

```

**Table 4: Arc-Path LP Formulation of the Lower Bound Model**

Like the standard deterministic throughput model, the model for the lower bound summarizes the flow constraints with an arc-path incidence matrix. There is no difference between the constraint set of this lower bound model and that of the deterministic model. The reliabilities of the network components come into play in the objective function where the optimal path flows are weighted by the path reliabilities. The path reliabilities are simply a product of component arc reliabilities since arcs are, by definition, connected in series to form a path. The upper bound is developed by similarly modifying the deterministic model to reflect the stochastic nature of the network components and overestimate the expected max flow. The upper bound model is shown in Table 5. For this model, the objective function remains unchanged from the deterministic case while the constraint set is changed. Here, the

```

Model Parameters:
   $u_i$  -- the capacity of arc 'i'
  A -- the arc-path incidence matrix
   $r_i$  -- the reliability of arc 'i'
   $e(u_i)$  -- the expected capacity of arc 'i'

Decision Variables:
   $f_j$  -- the flow across path 'j'

Model Definition:
  MAX  $\sum f_j$  -- maximize the sum of path flows
  ST  $Af \leq e(u)$  -- expected arc flow capacities
      $f_j \geq 0$  -- only positive path flows

```

**Table 5: Arc-Path LP Formulation of the Upper Bound Model**

arc capacities are replaced with expected arc capacities. These bounding models present a computationally feasible alternative to total network state enumeration.

**2.2.3 Network Reliability.** Another measure used for stochastic network performance estimation is reliability (digital transmission system *availability*) which measures probabilistic connectivity between a source node and a sink node in a network. Similarly, there are versions of reliability that find the probability that all nodes can communicate with all nodes or that all nodes can communicate with a designated source node [7:824]. In most cases, the computation of reliability is simplified by finding bounds instead of exact values and also by assuming that only arcs are subject to failure. Failing nodes may be replaced by a pair of perfectly reliable nodes connected with a failing arc. Bulka [9] presents a method that does not require the assumption of perfectly reliable nodes. His method extends the reliability measure to two dimensions, one for arcs and one for nodes.

The brute force method for computing the exact source to sink reliability in a network is to enumerate all possible network states and then sum the probabilities associated with each state where the source and sink are connected. Short of resorting to bounding the exact reliability, there are some techniques that compute source to sink network reliability that are

considerably more efficient than state enumeration. Jansen [17:20] examines three such techniques: breaking path reliabilities down into a set of disjoint products, factoring the total reliability value, and reducing the statespace to a set of most probable states.

The method of disjoint products is used to remove all dependence from the set of path reliabilities for a given network. The sum of all of the resulting disjoint product terms is the exact value for source to sink reliability. Abraham [1] presents an algorithm that uses boolean algebra to obtain the set of disjoint products for a network. This algorithm is improved upon by Aggarwal [3] who reduces the computational effort required by eliminating some intermediate steps. Further improvements are made by Locks [21] by using boolean minimization and rapid inversion to achieve substantial computational savings in processing large networks. Abraham's basic algorithm is summarized in Table 6.

```

Boolean Algebra Notation:
a    -- the number of arcs in the network
xi  -- boolean variable for edge i status
n    -- the number of source to sink paths
 $\phi$   -- the null set
Si  -- the boolean product of x's for a simple path
Pi  -- the boolean product which implies some Si
PDk -- the set {pj} derived from Sk

COMPARE Procedure:
1. Set X= $\phi$ 
2. Do Step 3 for all m where 1 ≤ m ≤ a
3. If variable m is 1 in Sj and 0 in Pi, Return, Disjoint
   If variable m is 1 in Sj and - in Pi, Put m in X

Main Algorithm:
1. (Initialize) DISJOINT SET = {S1}
2. Do Steps 3-6 for all k where 2 ≤ k ≤ n
3. Let PDk = {Sk}
4. Do Step 5 for all j where 1 ≤ j ≤ k-1
5. For each Pi in PDk, call COMPARE(Pi, Sj);
   If disjoint, repeat for the next Pi;
   If X= $\phi$ , drop Pi and repeat for next Pi;
   Replace Pi in PDk with the set of product terms,
   Repeat for the next Pi.

```

**Table 6: Abraham's Boolean Algebra Algorithm for Disjoint Products**

Abraham's original algorithm as presented in Table 6 computes an exact value for network reliability. This algorithm, while significantly more efficient than complete state enumeration, is not computationally feasible for evaluation of large computer networks. Heidtmann [15] presents what is probably the fastest disjoint product algorithm. This algorithm makes use of subproduct inversion to consider multiple paths simultaneously which speeds up computation time considerably, but even this algorithm remains NP-hard.

The reliability algorithms that use factoring are based on the principle that the reliability of a system may be broken down into the sum of the reliability of the system given a certain component is functional and the reliability of the system given the same component is not functional [30:269]. This is illustrated by the following equation:

$$R(S) = P(x_i=1)R(S|x_i=1) + P(x_i=0)R(S|x_i=0)$$

This concept may be employed recursively to decompose the exact reliability of a given network. Page and Perry [24] have put together a computer program that incorporates the concept of factoring with network reduction strategies to compute network reliability very efficiently. Some potentially useful network reduction strategies include replacing multiple components in series or parallel with a single component, eliminating unnecessary components, and possibly solving subnetworks individually ahead of time [30:272-273].

Both the disjoint product methods and the factoring methods involve finding efficient methods to solve the computationally difficult problem of calculating an exact value for network reliability. The third technique to be discussed here, most probable state enumeration, is used to bound the exact value for network reliability. Typically, there exist a small set of states in which the system operates most of the time. These states are used to obtain upper and lower bounds on the expected reliability of the system thus avoiding the "statespace

explosion" inherent in computing exact reliability for large networks [20:1105]. Among the most probable states that are considered for the bounds, those states where there is a functional path between source and sink are considered connected states while the rest are considered disconnected states. The lower bound is obtained by calculating the sum of the probabilities of the connected states and the upper bound is one minus the sum of the probabilities of the disconnected states.

Several algorithms have been developed to generate, in order, the most probable states for a network [20,28]. This approach seeks to find the most probable states, one at a time, starting with the most probable and working down from there, until the sum of the probabilities of all states found reaches a certain threshold. It is easy to see that the higher that threshold value, the more states that will be included, and the tighter the bounds on exact reliability. It is apparent that there is an inherent tradeoff between tight bounds and computational effort. The process of finding the set of most probable states can be simplified if the assumption is made that all components are highly reliable (as in most communication networks). If this assumption is made, the process is reduced to choosing the maximum number of failed components a state may have and still be included in the most probable set. In this way a value  $n$  is found such that the sum of the probabilities of all states with  $n$  or less failed components reaches or exceeds the probability threshold.

## ***2.3 Dynamic Networks***

***2.3.1 Stochastic Programming.*** The performance of a communications network will vary over time due to the variability in the performance of individual components with respect to routine and unexpected down time. Effort has been made recently to represent networks as

stochastic programming problems in order to capture the temporal aspects of network performance [12,14,25]. In this sort of an approach, the nodes in the networks are represented as a set of queues and the capacities are random variables. Frantzeskakis [14] presents some procedures for bounding performance using three different methods: an adaptation of Jensen's inequality, Monte-Carlo Simulations, and an analytical approximation procedure. This approach is typically not feasible to solve in a closed form and lends itself well to the application of simulation analysis.

**2.3.2 Dynamic Network Flow.** Dynamic network flow models are used for problems where there is an underlying temporal dimension such as the flow of information between nodes over time in a computer communication network. In essence, there are multiple copies of the original network, one for each point in time, and the copies are connected temporally by the arcs. This construct is referred to as a time-expanded replica of the original network and has a period corresponding to the interval of time the replica covers. Traditionally, this network model is used to find maximum network flow in terms of the maximum number of flow units that can go from source to sink in a specified time period as opposed to the maximum per unit time flow found in a static model. Maximum dynamic network flow can be found by solving the standard static max flow problem for a time-expanded replica of the network in question expanded to represent the correct period [4:737-738]. It should be noted here that the interval maximum flow value divided by the period gives a per unit time flow measure similar to the standard static maximum flow value. For finite periods, this surrogate maximum flow value will be significantly less than the static max flow value, but they two separate measures will converge as the period evaluated gets arbitrarily large.



```

Model Parameters:
  c(x,y)  -- the capacity of arc (x,y)
  a(x,y)  -- the traversal time of arc (i,j)
  P  -- the period of the dynamic flow

Decision Variables:
  f(x,y;τ) -- flow leaving x along (x,y) at time τ
  f(x,x;τ) -- hold over at x from time τ to τ+1
  V(P)  -- net source to sink flow in p time units

Model Definition:
  MAX V(P)
  ST   $\sum_y [f(s,y;\tau) - f(y,s;\tau - a(y,s))] - V(P) = 0$ 
       $\sum_y [f(x,y;\tau) - f(y,x;\tau - a(y,x))] = 0 \quad (x \neq s, \tau = 0, 1, \dots, p)$ 
       $\sum_y [f(t,y;\tau) - f(y,t;\tau - a(y,t))] + V(P) = 0$ 
       $0 \leq f(x,y;\tau) \leq c(x,y)$ 

```

**Table 7: Maximum Dynamic Flow Model Using Time-Expanded Network**

Table 7 shows a linear programming representation of the maximum dynamic flow model that makes use of the time-expanded network representation. It is readily apparent that as the period to evaluate gets larger, the size of the equivalent time-expanded max flow model grows also, and eventually will not be feasible to solve for. For this reason, it is necessary to use a more efficient model that makes use of the concept of temporally repeated flows. A temporally repeated flow is a set of steady-state path flows. It has been shown that for any network there exists a temporally repeated flow that is maximal over all possible flows for that network [13:147-149]. Table 8 shows how this idea is used to formulate a model for max dynamic network flow that does not grow with respect to the period to be evaluated. The feasible region for this model is the same as that for a standard static max flow problem, but the objective function is unique. The objective function is formulated to reflect how often a given path would be repeated in a time-expanded replica based on the time it takes to flow from source to sink on the path and the period in question. It is obvious that this model is much more efficient than using the time-expanded replica model. It should be noted that this

```

Model Parameters:
  ui -- the capacity of arc 'i'
  A -- the arc-path incidence matrix
  σj -- the traversal time of path 'j'
  P -- the period of the dynamic flow

Decision Variables:
  fj -- the flow across path 'j'

Model Definition:
  MAX  $\sum (P+1-\sigma_j) f_j$  -- sum temporally repeated path flows
  ST  Af ≤ u               -- arc flow capacities
      fj ≥ 0               -- only positive path flows

```

**Table 8: More Efficient Maximum Dynamic Flow Model**

framework for finding maximum dynamic flow (throughput) could also be applied to calculating other dynamic network performance measures.

## 2.4 Summarizing Vector Time-Series

**2.4.1 MARMA Models.** Monitoring multiple network performance measures over time will yield a set of vector time-series performance data and it is necessary to summarize the information presented by this data. It is expected that there will be multivariate correlation between the separate measures as well as temporal correlation within each univariate time-series. A multivariate autoregressive moving average (MARMA) model could be fit to a set of performance data to summarize the effects of both types of correlation in the system [11]. MARMA is the multivariate extension of an ARMA model which combines an autoregressive model which attempts to describe the process as a weighted sum of current and previous observations, and a moving average model which attempts to describe the process as a weighted sum of previous error terms. The extension to multivariate analysis attempts to gain additional explanatory power by adding a weighted influence from the other performance measures.

**2.4.2 Simultaneous Estimating Equations.** The same correlations discussed previously may be represented by a set of simultaneous estimating equations. Each of the estimating equations contains a previous observation term to capture the temporal correlation and also a term for the current observation of all other univariate time-series measures to capture the multivariate correlation. Below is an example of estimating equations for vector time-series network performance in terms of throughput (T) and reliability (R):

$$T_{t+1} = a(R_{t+1}) + b(T_t) \quad \text{and} \quad R_{t+1} = a'(T_{t+1}) + b'(R_t)$$

In this case, each estimating equation has one exogenous variable, the lagged observation, and one endogenous variable, the current observation of the other performance measure. It is possible to calibrate this sort of an equation using a process called two-stage least squares [11]. The first step in this process is to generate estimates for the endogenous variables by running regression models for each equation using only the exogenous variables. Then the full models are regressed using the exogenous variables and the previously generated estimates for the endogenous variables. A set of equations like this could be used to summarize one set of vector time-series data or make comparisons between multiple data sets.

### ***III. Methodology***

In this chapter, the overall research plan that is followed to accomplish the specific research objectives given in Chapter I is discussed. First, models are presented for quantifying measures for both absolute network performance and relative network performance. Then, an algorithm for path enumeration is presented along with various software tools, both of which are necessary in the application of the previously defined performance models.

#### ***3.1 Understanding the Problem***

The sponsor of this research seeks to identify appropriate measures of performance and techniques for proactively monitoring these measures in a computer communications network. This effort will focus on identifying and modeling some promising measures of performance while a parallel research effort will identify potentially useful monitoring techniques. Together, the two efforts will present some possible solutions to the problem of proactively monitoring network performance. By itself, this work demonstrates some useful tools for investigating stochastic network performance.

In order for a performance measure to be useful in proactive network monitoring, it must reflect both the stochastic and temporal nature of an operational computer communication network. The building blocks of this research effort are the network performance models presented by Jansen [17] and Yim [31]. These models were developed to place bounds on the expected values for throughput and reliability in stochastic networks. Although the performance measures quantified by these models do capture the stochastic nature of the networks in question, the temporal dimension is missing, and must be included in

the models. Also, this sort of performance measurement is not directly applicable to the proactive monitoring problem because these are measures of absolute network performance (network performance at maximum utilization). Since actual network performance will be relative to the level of network utilization at a given time, it is necessary to formulate models for measures of relative network performance.

To make the move away from static, absolute performance measure models to dynamic, relative performance measure models, it is necessary to extend the previously presented maximum stochastic network flow bounding models. The static max flow bounding models are transformed to dynamic max flow bounding models and then to dynamic minimum cost flow models. It is important to show that as these transitions are made, the models still bound the expected network flow.

The original bounding models proposed by Yim [31] are based on an extension of Jensen's inequality. The maximum network flow is a function the capacities of the arcs in the minimum cut set for the network. This is illustrated in the following equation where  $t$  is the maximum network flow and the  $X_i$ 's are the arc capacities.

$$t = f(\mathbf{X}) = f(X_1) + f(X_2) + \dots + f(X_n)$$

It can be shown that this function is concave with respect to each of the arc capacities since the function will either increase linearly or remain constant as an arc capacity is increased. Given that the maximum network flow function is concave, Jensen's inequality can be applied to show that the following relationship exists.

$$E[f(\mathbf{X})] \leq f(E[\mathbf{X}])$$

This relationship suggests that the maximum expected flow must be less than the maximum network flow when arc capacities are set at their expected values. This relationship is the

basis for the model that computes the upper bound on the expected maximum network flow.

Jensen's inequality is extended to show the following relationship.

$$\{E[f(X_1)] + E[f(X_2)] + \dots + E[f(X_n)]\} \leq E[f(X)] \leq f(E[X])$$

This relationship suggests that the sum of the expected flows on arcs in the minimum cut set must be less than the maximum expected flow. This relationship is the basis for the model for the model that computes the lower bound on the expected maximum network flow.

The first extension of these expected maximum network flow bounding models is the move to a dynamic network flow representation. The only difference between the static and dynamic maximum flow problems is the value of the parameters in the objective functions. With this in mind, it is apparent that the maximum dynamic network flow is still a concave function of the capacity of the cut set arcs and the relationships shown with Jensen's inequality still exist. This confirms that the models may be extended to the dynamic network flow case and still bound the expected maximum flow.

The extension to the minimum cost network flow problem requires that the new objective function is shown to be concave so that Jensen's inequality still holds. When the transition is made from max flow to min cost, the original objective function becomes a constraint and is replaced by a new function of arc capacities representing flow cost. The form of the new objective function and constraint are shown below.

$$\text{MIN } c_1X_1 + c_2X_2 + \dots + c_nX_n$$

$$\text{ST } X_1 + X_2 + \dots + X_n = F$$

This is a minimization problem so the objective function must be shown to be convex. The corresponding maximization problem's objective function will be concave and Jensen's inequality will hold.

To determine if this function is concave with respect to the individual arc capacities the function's behavior is studied as an arc's capacity is increased. If modifying the overall flow to distribute more flow along the arc in question can improve the objective function, flow is shifted from the most costly arc in use to the arc with the recently improved capacity. In this way, the minimum cost function decreases linearly until all flow on the expensive arc has been reassigned to the new arc at which point flow is either shifted from another expensive arc, linearly decreasing the objective function at a slower rate than before, or flow can not be improved and remains constant. This behavior describes a piece-wise linear cost function that decreases at a decreasing rate with respect to arc capacities. Obviously, an objective function of this type is convex with respect to arc capacity and the corresponding maximization problem objective function ( $\text{MIN } Z \rightarrow \text{MAX } -Z$ ) is concave.

Now that the minimum cost model objective function has been shown to be concave, the previously described relationship derived from Jensen's inequality is applicable and the original flow bounding technique may be used. This allows upper and lower bound estimates to be computed for expected arc or path flows given a set level of overall source to sink flow. The difference between the set of optimal flows at the lower bound and those at the upper bound represents a shift towards using more costly paths. Cost here is a measure of the amount of time or hops to flow from source to sink along a given path. If it is assumed that the more costly paths are also less reliable, since they are longer and have more components that can fail, then the two relative performance measures that will be modelled later, BER and delay, may both be expected to increase linearly between the lower and upper bound path flows. This guarantees that the actual relative performance will be bounded by the values computed at the lower and upper bound solutions for the minimum cost flow.

### 3.2 Modelling Absolute Performance Measures

**3.2.1 Absolute Performance.** An absolute performance measure is defined here as one that quantifies some aspect of network performance without regard to the specific level of network utilization. Any measure of network performance strictly at maximum flow or unrelated to the amount of flow would fall into this category. The throughput and reliability bounding models presented in chapter II fall into this category. As observed previously, these models already account for the stochastic nature of the problem, but fail to capture the temporal dimension.

**3.2.2 Network Throughput.** The absolute performance measures for quantifying network throughput are based on the arc-path formulations for upper and lower bounds on expected maximum flow that were presented in Chapter II. To incorporate the temporal aspect of the problem, these models are extended to a maximum dynamic flow format also given in chapter II. Tables 9 and 10 show the resulting dynamic network flow models for lower and upper bounds on expected maximum flow.

```
Model Parameters:
  ui -- the capacity of arc 'i'
  A -- the arc-path incidence matrix
  Rj -- the reliability of path 'j'
  σj -- the traversal time of path 'j'
  P -- the period of the dynamic flow

Decision Variables:
  fj -- the flow across path 'j'

Model Definition:
  MAX  $\sum (P+1-\sigma_j)R_j f_j$  -- weighted sum of path flows
  ST  Af ≤ u               -- arc flow capacity constraints
      f ≥ 0                -- non-negative flow constraints
```

**Table 9: Max Dynamic Flow Lower Bound Model Formulation**



```

Model Parameters:
  ui -- the capacity of arc 'i'
  ri -- the reliability of arc 'i'
  e(ui) -- the expected capacity of arc 'i'
  A -- the arc-path incidence matrix
  σj -- the traversal time of path 'j'
  P -- the period of the dynamic flow

Decision Variables:
  fj -- the flow across path 'j'

Model Definition:
  MAX  $\sum (P+1-\sigma_j) f_j$  -- temporally repeated path flow sum
  ST  Af ≤ e(u)          -- arc flow capacity constraints
      f ≥ 0              -- non-negative flow constraints

```

**Table 10: Max Dynamic Flow Upper Bound Model Formulation**

As seen in the two model formulations, the reduced form of the dynamic network flow model was used. Not only does this greatly reduce the computational effort required to solve the dynamic flow models for large networks, but the idea of temporally repeated path flows is useful in analyzing the performance of a network. This formulation makes use of the feature of dynamic network flows that specifies that there will always exist a temporally repeated network flow that is maximal over all possible dynamic network flows. Due to this feature, the series of flows across a given path in a network over time may be reduced to a single steady-state type flow. Given single values for the dynamic network flow on each path, the process of identifying critical paths is simplified. Note that this useful feature is limited by the fact that this is an absolute performance measure. This means that the information obtained from study of these path flows applies only to the network when operating at maximum utilization.

**3.2.3 Network Reliability.** Most probable state enumeration is used here for the development of an absolute measure of network reliability. Factoring, disjoint products, and complete state enumeration, presented in the reliability section of chapter II, are not used here because each of these techniques become computationally infeasible when attempting to analyze large networks. With most probable state enumeration, there is an implicit trade-off between the degree of computational effort and obtaining a tight bound on the exact reliability.

Components within a computer communications network are generally highly reliable. This fact leads to the conclusion that the most probable states are those where only a few of the network components are in a failed state, and this allows the process of enumerating the most probable states to be simplified. This simplification is done by finding a constant  $k$  such that the sum of the probabilities of all states with  $k$  or fewer failed components exceeds a specified threshold probability. For instance, it might be found that for a network with 50 components, the sum of the probability of all states with 5 or fewer failed components exceeds the goal of 95% cumulative probability. In this way, a small subset probabilistically dominates the rest of the statespace and this subset can be used to obtain bounds on the exact value for network reliability.

This method of statespace reduction could be applied to a static network or a time-expanded network to allow for temporal modelling. At this point, all that remains is to evaluate each state in this reduced set to determine upper and lower bounds on the exact network reliability. Table 11 shows the functional form for each of these network reliability bounds.

Model Parameters:	
MP	-- the set of 'Most Probable States'
$P_i$	-- the probability of being in state 'i'
$S_i$	-- the status of state 'i' (If S-T is connected then 1, else 0)
Lower Bound Reliability Formulation:	
	$\sum_{i \in MP} P_i S_i$
Upper Bound Reliability Formulation:	
	$1 - \sum_{i \in MP} P_i (1 - S_i)$

**Table 11: Bounding Reliability With the Most Probable States**

The key to this method is highly reliable components leading to the most probable states being those with very few failures. As this assumption becomes less valid, the constant  $k$  will become larger, and so will the size of the statespace subset necessary to probabalistically dominate the whole. If the subset grows to large, computational feasibility comes into question again and a more efficient method would be required for enumerating the most probable states. For now, it is assumed that this assumption is valid and the above described process may be applied to obtain bounds for network reliability.

Like the throughput models discussed previously, this measure of performance has no temporal aspect. To extend the fomulation temporally, either average periodic static reliability estimates, or use a time-expanded network to obtain an interval reliability estimate. The latter method is of questionable value since it would represent the probability of having any source to sink connectivity during an interval specified by the period. This measure would approach unitary value as the period gets large.

### **3.3 Modelling Relative Performance Measures**

**3.3.1 Relative Performance.** A relative performance measure is defined here as one that quantifies some aspect of network performance with respect to the degree to which the network is being utilized. Any measure of network performance that has a different value for the same network, in the same state, depending on the amount of source to sink flow, will be considered a relative performance measure. These are the type of performance measures that might be monitored real-time on an operational computer communications network. In this section, models for bit error rate and delay are presented as candidates for real-time proactive monitoring. The models will be structured as minimum cost flow problems rather than the maximum flow type problems presented earlier. These relative performance measures must capture the stochastic and temporal aspects of communication networks as discussed previously.

**3.3.2 Bit Error Rate (BER).** Bit error rate in a digital communications system is the ratio of the number of bits received at the sink to the total number of bits sent out from the source, assuming a single source and sink. The data bits sent from the source that are lost and never reach the sink are called error bits and BER is just one of the various performance measures that deal with sort of data loss. Development of the BER model starts at the dynamic network flow bounding models presented previously. In both the upper and lower bound models, the objective is to maximize total source to sink flow while maintaining certain network constraints. For the BER models, the network constraints remain the same, but source to sink flow is a constant and the former objective function is added to the constraint set. The new objective is to minimize total cost in terms of either traversal time or perhaps more appropriately, number of *hops*. This follows the assumption in chapter I that all routing

```

Model Parameters:
ui  -- the capacity of arc 'i'
ri  -- the reliability of arc 'i'
e(ui) -- the expected capacity of arc 'i'
A    -- the arc-path incidence matrix
Rj  -- the reliability of path 'j'
σj  -- the traversal time of path 'j'
P    -- the period of the dynamic flow
F    -- Level of source to sink flow

Lower Bound Decision Variables:
Rjfj -- the flow that reaches sink over path 'j'

Lower Bound Model Definition:
MIN  $\sum (P+1-\sigma_j) \sigma_j R_j f_j$  -- MIN cost for flow level F
ST   $\sum (P+1-\sigma_j) R_j f_j = F$  -- S-T flow level constraint
    Af ≤ u                    -- arc flow capacity constraints
    f ≥ 0                     -- non-negative flow constraints

Upper Bound Decision Variables:
fj  -- the flow that reaches sink over path 'j'

Upper Bound Model Definition:
MIN  $\sum (P+1-\sigma_j) \sigma_j f_j$  -- MIN cost for flow level F
ST   $\sum (P+1-\sigma_j) f_j = F$  -- S-T flow level constraint
    Af ≤ e(u)                 -- arc flow capacity constraints
    f ≥ 0                     -- non-negative flow constraints

```

**Table 12: Upper and Lower Bound Models For Min Cost Flow**

is done using a shortest path algorithm. Table 12 illustrates the conversion of the max dynamic flow problems to min cost dynamic flow problems.

The decision variables for the two models shown in Table 12 give upper and lower bounds on the expected amount of flow to reach the destination node over each path. These values represent the non-errored bits for each path repeated temporally during each period. To obtain estimates for BER from this point, each path reliability can be used as a gain (loss) factor. Give the number of non-errored bits for each path and a gain factor, the total number of bits sent along each path may be obtained. Values for total bits sent and non-errored bits are used to obtain values for errored bits and BER. The mechanics are shown in Table 13.

Model Parameters:

$R_j$  -- the reliability (gain) for path 'j'  
 $S_j^l$  -- non-errored bits for path 'j' (lower bound)  
 $S_j^u$  -- non-errored bits for path 'j' (upper bound)

Calculations:

$T_j^l = S_j^l / R_j$  -- total bits sent for path 'j' (lower bound)  
 $T_j^u = S_j^u / R_j$  -- total bits sent for path 'j' (upper bound)  
 $E_j^l = T_j^l - S_j^l$  -- errored bits for path 'j' (lower bound)  
 $E_j^u = T_j^u - S_j^u$  -- errored bits for path 'j' (upper bound)

Bounds on Expected BER:

Lower Bound BER --  $\sum (E_j^l) / \sum (T_j^l)$   
Upper Bound BER --  $\sum (E_j^u) / \sum (T_j^u)$

**Table 13: Bounding BER From Min Cost Flow Models**

The two values that are calculated using the procedure described in Table 13 are bounds on the expected BER for a given, constant level of source to sink flow. These bounds will vary depending on the level of flow and, in general, BER will be higher for higher levels of flow (network utilization).

**3.3.3 Delay.** Delay is another relative network performance measure that might be monitored on an operational computer communications system. Delay is a measure of the average length of time it is taking to transmit one data bit from source to sink in a network. The level of network utilization is represented here by a constant value for network flow, and as this value rises, so does the delay. For this reason, delay fits into the previously defined category of relative performance measures and can only be estimated relative to a given utilization level.

The models for bounding the expected value of the delay performance measure follow directly from the min cost dynamic flow models developed previously for BER. The decision variables for these min cost models were the non-errored bits broken down by path. The traversal time for each path has already been defined as a parameter in these models. Using

Model Parameters:

$\sigma_j$  -- the traversal time of path 'j'  
 $S_j^l$  -- non-errored bits for path 'j' (lower bound)  
 $S_j^u$  -- non-errored bits for path 'j' (upper bound)

Calculations for Bounding Expected Delay:

Lower Bound --  $\sum [(P+1-\sigma_j) \sigma_j S_j^l] / \sum [(P+1-\sigma_j) S_j^l]$   
Upper Bound --  $\sum [(P+1-\sigma_j) \sigma_j S_j^u] / \sum [(P+1-\sigma_j) S_j^u]$

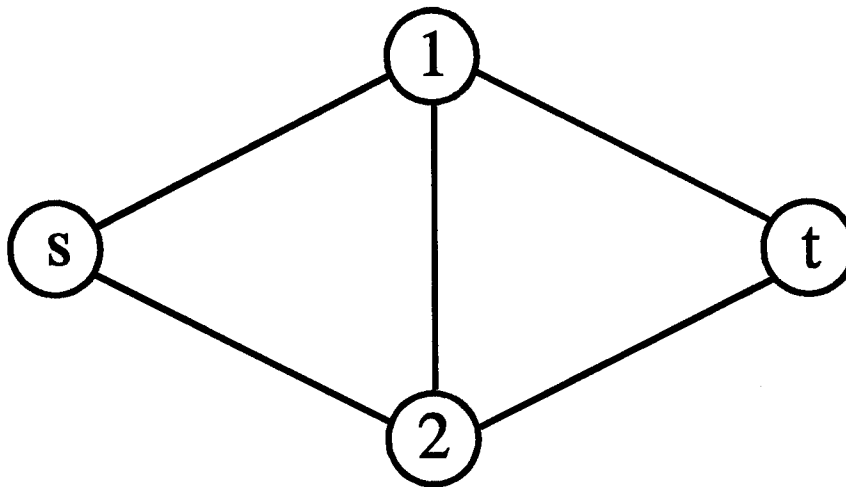
**Table 14: Bounding the Average Delay Performance Measure**

the values for non-errored bits and traversal time for each path, upper and lower bounds on the expected delay are obtained as shown in Table 14. Note that the delay bounding values may represent actual traversal time or number of 'hops' depending on the values chosen for the model parameters.

### 3.4 Path Enumeration Algorithm

All of the performance measures presented so far are obtained from the optimal solution to either a max flow or min cost flow model formulated with an arc-path incidence matrix for the constraints. This is useful in that optimal network flows may be decomposed into a set of optimal path flows, but this sort of model formulation requires all source to sink paths to be identified before any performance measure model can be constructed. While manual path enumeration may be a trivial task for networks of just a few nodes, large network structures pose a more challenging problem. For this reason, an algorithm is presented here to accomplish the task of complete enumeration of all source to sink paths. This algorithm represents a network as a tree structure and performs a depth-first exhaustive search of all branches that, when implemented in a structured programming language, automates the difficult task of path enumeration. To illustrate the depth-first search method, Figure 1 shows a sample network and the associated tree structure to be searched.

## Network Representation



## Tree Representation

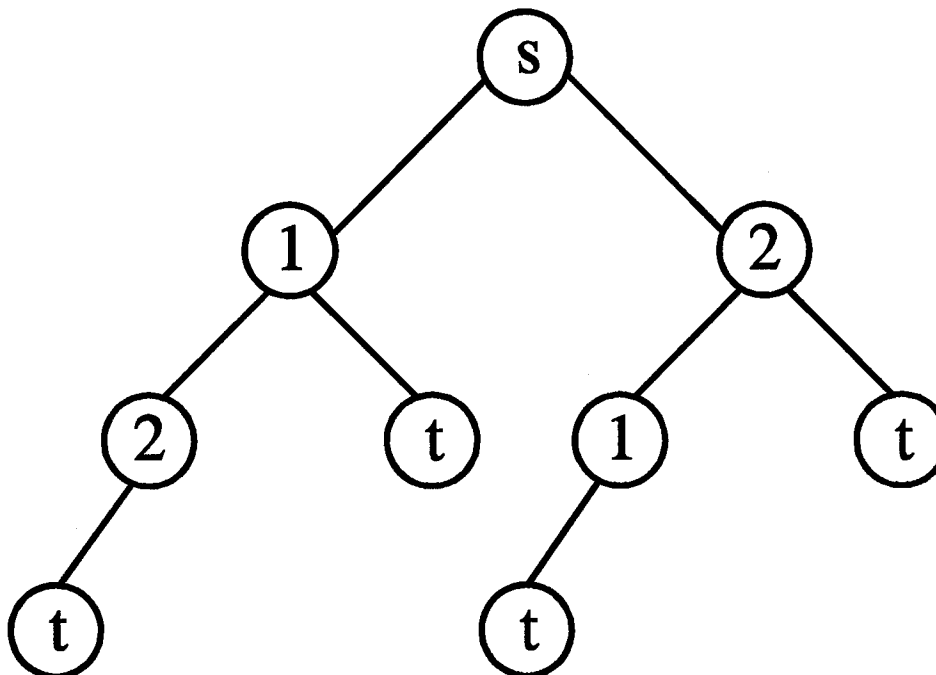


Figure 1: Representing a Network as a Tree Structure



The small sample network in Figure 1 consists of a source node, two intermediate nodes, and a sink node. This network is fully connected and it is trivial to manually enumerate the four paths from source to sink. The tree seen in this figure is generated by branching, starting with the source node, to all connected nodes. Every branch is extended until the sink node is reached or a node is reached where it is not possible to branch to a node not already on the branch. The search algorithm considers the left-most branch first and if it terminates at the sink it is a valid path and is added to the set of source to sink paths. In this manner, branches are evaluated until the left-most branch is considered, at which point the set of paths is complete.

## ***IV. Implementation***

In this chapter, the approach taken to automate the methodology presented in Chapter III in a computer implementation of the modelling process is discussed. First, a presentation of software selection alternatives and rationale for the choices that were made is given, followed by a discussion of the program design that breaks down and describes each of the major program components. Then, program control is described, detailing how the program components are linked together to accomplish the goals of the implementation. Finally, a summary is given to outline the intended use of the program to demonstrate the proposed methodology.

### ***4.1 Software Selection Decisions***

The major components of this implementation include input, output, path enumeration, model construction, model solution, and control. Previous work in this area made use of the artificial intelligence programming language Prolog to perform the path enumeration and model construction tasks [17,31]. Prolog is particularly well suited to prototype implementation of tree search type algorithms like path enumeration because it can manipulate linked list data structures handily without a great deal of overhead lines of code to manage them. While language selection here may affect ease of implementation, or perhaps actual run time, the efficiency (number of steps necessary to perform the algorithm) is constant with regards to programming language.

Selection of a software option to perform the solution of the models is a more significant issue. Here, software selection could substantially affect the solution time for large network models. Network problems have certain unique features that allow for the use of more efficient solution algorithms than standard linear program solvers. The savings from these faster algorithms grow with the size of the network in question such that eventually, solution with a standard solver may become infeasible.

The sponsor described a typical network of concern would have up to 50 nodes and each node might have as many as 10 emanating arcs. Experimentation was performed with sample networks in this size range using different solution algorithms to get a feel for the differences in efficiency, and hence in solution time. Some software options considered include SAS Netflow (network solver), IMSL (revised simplex), and CPLEX (both standard and simplex solvers. CPLEX was particularly useful in comparing the efficiency of standard and network specific solvers since a problem could be loaded then solved with each solver in turn. This software package even presented a breakdown of solution time after solving a model with a given solver. The results of this experimentation showed that, for the size of networks in question here, differences in efficiency were small enough that standard solvers do not need to be eliminated from consideration.

Since no path enumeration, model construction, or model solution considerations demand any particular software implementation, implementation decisions are made to allow for the most efficient input, output, and control. Some of the applications of this program are to iteratively modify and resolve a given model formulation. For this reason, it is most efficient to handle all implementation components within a single program, unlike previous efforts where components were separate programs and data was manually exchanged.

With the main concern being to efficiently handle all implementation tasks inside of a single program, FORTRAN and IMSL were chosen for implementation software. The path enumeration algorithm is more difficult to implement in FORTRAN, with some extra overhead to maintain the linked lists, but there were no problems with encountered as far as solution time. All of the models called for by the methodology are constructed internal to a single Fortran executable program using the set of enumerated network paths. At compile time the executable is linked to the set of IMSL subroutines allowing for efficient vector and matrix manipulation as well as solution by one of several possible linear program solvers in IMSL. The major benefits of this particular implementation are one step input, internal control, and output easily organized and tailored to a workable format. Figure 2 illustrates the breakdown of the major components of this implementation all to be controlled within a single FORTRAN executable program.

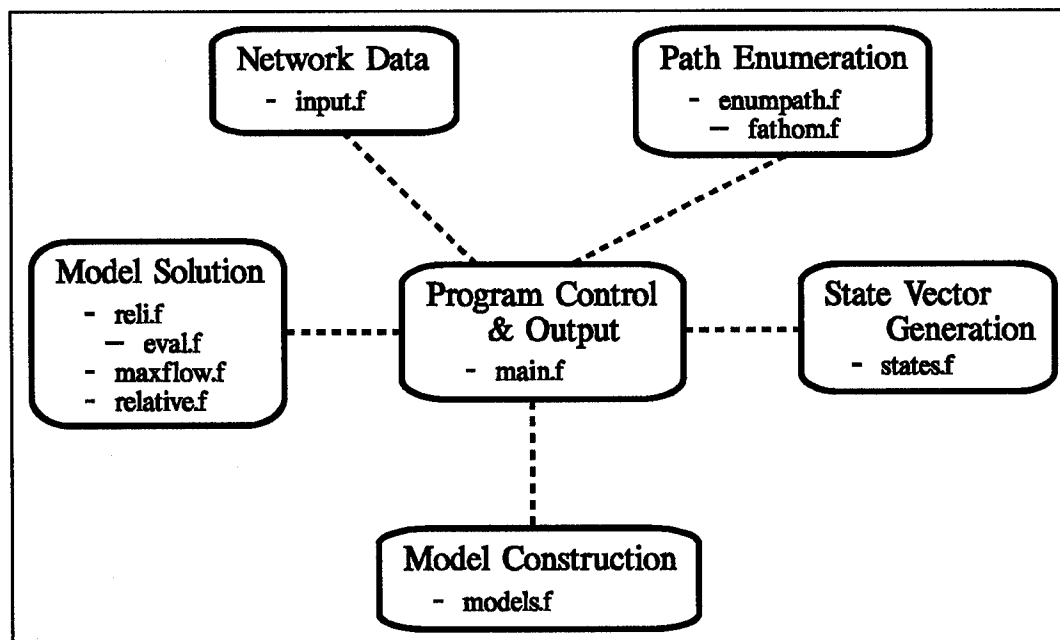


Figure 2: FORTRAN Implementation Components

## **4.2 Program Design**

**4.2.1 Input.** The design for the program to implement this methodology calls for input describing the network in question at start time and later, input specifying some model parameters. The start time input consists of a single file containing values for all of the parameters necessary to describe a network to be analyzed. This file is read into data structures in the program at run time and must contain a line containing the number of nodes, one for the number of arcs, and one each for the parameters of the arcs. Each arc parameter line in the network description file gives values for the arc's origin node, destination node, reliability, capacity, and traversal time (set equal to one if counting 'hops'). The network description file must be in a precise format to be read into the program and this format is demonstrated in the sample network description file in Appendix A.

The other type of input necessary for this implementation is prompted for during program execution. Input necessary here includes the period to use in formulating and evaluating the dynamic network performance models, the utilization level for the relative performance models, and the number of observations to compute when estimating vector time-series performance data. The period selected needs to be longer than the length of the longest source to sink path in the network. The utilization level is a constant level of source to sink flow and should be set with regards to the lower bound on the expected maximum flow in order to avoid feasibility problems in the relative performance models. The number of observations value is only limited by a constant dimensioning parameter in the code that may be changed if necessary. This necessary manual input is considerably less manual intervention than what would be needed if the model construction and model solution components were implemented as separate executables using separate software options.

**4.2.2 Path Enumeration.** The task of enumerating all of the network's source to sink paths requires no further input than the single network description file read into memory at run time. The source node and sink node are designated implicitly by the numbering of the nodes within the network description file. Node one is assumed to be the source node and node  $n$  is taken as the sink node. This point must be accounted for when building the description file for any networks to be analyzed.

The path enumeration task is accomplished using one primary FORTRAN subroutine which in turn makes calls to another, secondary subroutine. The primary subroutine converts the network structure, that was read from the network description file, into a linked list, tree structure representation of the network. This subroutine then iteratively calls the secondary subroutine to extract complete branches from the tree structure and evaluate whether the branch constitutes a source to sink path. The set of all source to sink paths is completed in this manner by evaluating branches, working left to right, until every branch has been considered. Aside from building the path set, this implementation component also builds all other data structures containing path parameters necessary for model construction. These path parameters include the number of path elements, a value for path reliability, and the number of temporal path repetitions.

**4.2.3 Model Construction.** All of the model formulations from the methodology chapter that are implemented here are of the same general form. They are all either max flow or min cost network models formulated using an arc-path incidence matrix to express the arc capacity constraints. The parameter vectors and matrices for these models do vary significantly, but have many common elements. This component of the implementation attempts to exploit the common parameter elements while generating the minimum set of data

structures necessary to capture all of the parameter values for formulating and solving each of the individual performance models. This data is stored in a common block that is accessible to the model solution subroutines that send a different subset of the data to IMSL depending on which model is to be solved.

**4.2.4 State Vector Generation.** One task of this implementation is to use the causal models formulated to generate a set of vector time-series performance data to evaluate against actual network performance data to validate the methodology. In order to accomplish this task it is necessary to simulate the dynamic nature of an operational communication network. This is done here by randomly generating a series of binary state vectors that represent the up or down status of every arc in the system. Going back to one of the original assumptions made in the first chapter, all arcs are assumed to fail independently with a known probability. This probability is the arc reliability value obtained from the network description file.

The performance models to be evaluated here are dynamic arc-path formulations with path flow decision variables so arc state vectors alone are not sufficient. These 'static' arc state vectors are converted to 'dynamic' path state vectors. The up or down status of a given path at time  $t$  depends on whether a unit of flow leaving the source node along the path at time  $t$  will reach the sink node at time  $t + \sigma$ , where  $\sigma$  is the path traversal time. This depends on the status of each component arc of the path at the time the flow unit reaches that arc. Once the dynamic path state vectors are derived, the data is transformed a final time to come up with a new path state vector representing number of times a path fails in each interval of the specified period. The dynamic flow models all include a parameter for the number of temporal repetitions of each path in a period. This number is modified by the values in the path failure state vector for each interval so in this implementation component enough of these

state vectors need to be generated to estimate the number of observations called for by the user input. The resulting series of state vectors is passed back to the main program where one vector at a time is sent to the solver to iteratively estimate vector time-series performance data.

**4.2.5 Model Solution.** The model solution component of this implementation is probably the most straightforward of the components since there is not much original code necessary here. Three FORTRAN subroutines are used here to solve the reliability models, the throughput models, and the relative performance measure models. Each of these subroutines is passed a state vector to be used along with parameters from the common block generated previously as input to an IMSL subroutine call. The performance measures are computed from the optimal solution returned from IMSL and passed back to the main program.

The division of this task into three separate subroutines is based on the type of model being solved for each performance measure. The throughput measures call for a max flow model to be solved while the relative performance measure require a min cost solution and the reliability models are done with most probable state enumeration. This is a logical division and most of the work in the first two is handled by IMSL. The third subroutine, the reliability solver, enumerates the set of most probable states and then makes use of a secondary subroutine to evaluate each state for connectedness.

### **4.3 Program Control**

The application of the previously described subroutines to accomplish the goals of the program is handled in the main section of the FORTRAN code. The subroutines for reading



in the network description data and then enumerating the source to sink paths are called at the very beginning of this section of the code. These are only used this one time to initialize the data representation of the network in memory. The model construction subroutine must be called to build or rebuild the model parameters any time the period is changed for the dynamic flow models of the utilization level is changed for the relative performance measure models. Any time a set of vector time-series performance data is to be estimated, the state vector generation subroutine is run to build a set of network state vectors. The remaining subroutines are those for model solution and are called as necessary to bound performance or estimate time-series data. The subroutine calls within this section of code may be rearranged easily depending on the goal and formatted print or write statements may be used to get the output in a workable form. This is particularly useful when trying to obtain a data set ready for input into another software package for further analysis.

#### ***4.4 Summary***

While the implementation here is modular enough to be easy to modify to suit a specific purpose, for now it is set up for three main uses. The first intended application is for bounding all of the absolute and relative performance measures presented in the methodology chapter given a utilization level and a period for the dynamic flow. The second intended application is iterative estimation of the performance bounds with increasing period in order to show how the dynamic performance measure converge to the static values as the period grows arbitrarily large. The final intended application is the generation of estimated vector time-series performance data that is necessary for the validation of the models.

The appeal of this implementation here is the fact that all phases of the methodology

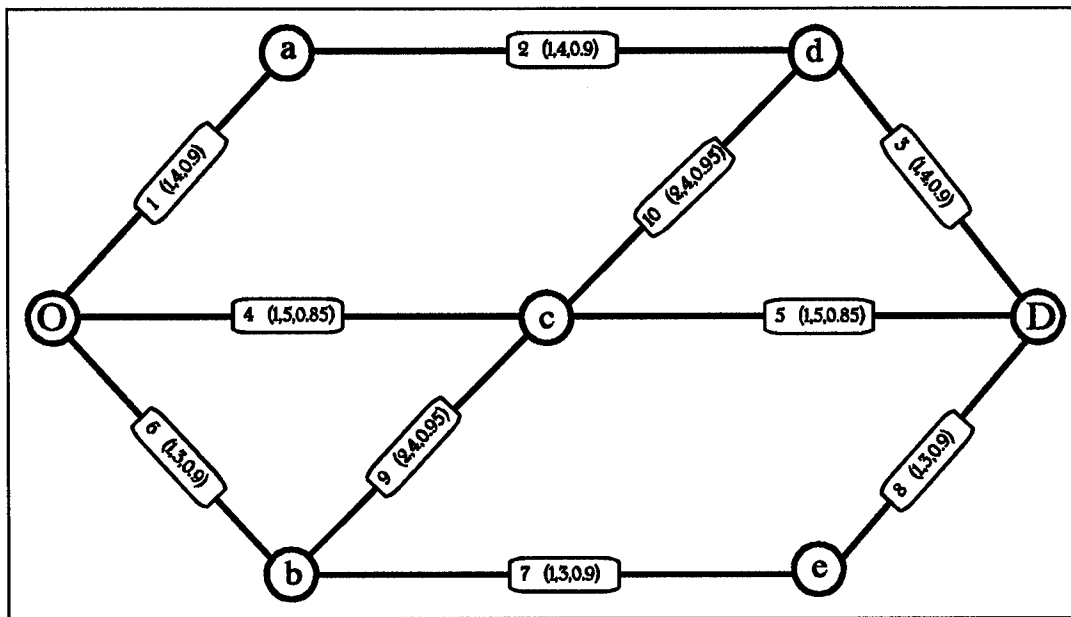
are handled inside one program. The drawback is that while the standard linear programming solver provided by IMSL was sufficient for the size networks looked at in this study, if large networks were to be studied, it might be necessary to go to a network specific solver for the sake of computational efficiency. If this were the case, the model solution subroutine used here is readily altered to write the model formulations to a problem file formatted for use with a solver like CPLEX instead of solving internally with IMSL. The cost would be the increased manual intervention to run each of the generated problem files through an external solver. This cost is acceptable when only attempting to bound performance, but significant when estimating vector time-series performance data observations.

## ***V. Results and Analysis***

This chapter demonstrates application of the methodology from Chapter III to bound the expected value of the dynamic network performance measures for some representative example networks. First, the mechanics of the methodology are demonstrated by constructing a small sample network and detailing the construction and solution of the models. Bounds are computed for the expected performance values of the sample network and then estimation of vector time-series performance data is demonstrated. Afterwards, a case study is performed by analyzing expected performance for three larger networks which are more realistic representations of communication network structure.

### ***5.1 Analyzing a Sample Network***

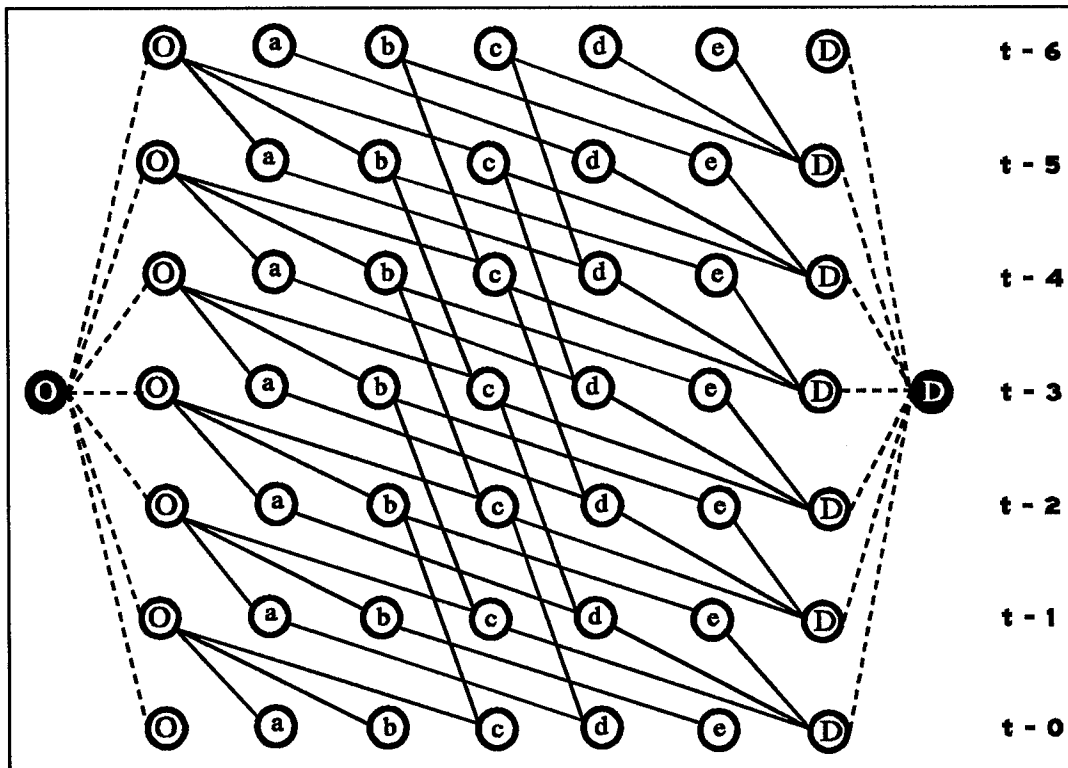
**5.1.1 Network Description.** To demonstrate the mechanics of the proposed methodology, a small sample network was constructed that follows all assumptions made previously, but is not necessarily representative of communication network structure. This network is illustrated in Figure 3. This sample network has an origin node, 5 intermediate nodes, and a destination node, a total of 7 nodes so the first line of the associated network description file holds the value 7. Likewise, the second line of the file would have the value 10 indicating the number of arcs in the network. By convention, the source node is referenced as node 1, the sink node is referenced as node 7, and the intermediate nodes are given arbitrary reference numbers. The network description file must contain 7 more lines, one for the parameters of each arc. Each arc line holds origination node, termination node, and the three arc parameters given in parentheses in the figure. For example, arc number 2, connecting



**Figure 3: Sample Network**

nodes 'a' and 'd' has a cost one time unit, a capacity of 4 flow units, and a reliability of 0.90. The ordering of the 7 arc parameter lines in the network description file has no significance and may be performed arbitrarily so long as all network arcs are represented.

The network representation illustrated in Figure 3 is a standard 'static' depiction of a stochastic network. The proposed methodology here calls for dynamic flow models to investigate network performance over fixed time intervals. This concept is demonstrated graphically with a construct known as a time-expanded network replica. Figure 4 shows one possible time-expanded replica of the sample network. Each row of nodes in the expanded network represents all of the original network nodes at a given time increment. Flow in this network is directed left to right, towards the super sink node, and arcs are directed downward one level for each time unit of cost for that arc. This time-expanded replica defines a dynamic period of 6 time units as seen by the seven rows of replicated node spanning 6 time units (see Figure 4).



**Figure 4: Time-Expanded Replica of the Sample Network**

In Figure 4 it is apparent that many of the arcs in the time-expanded sample network replica are not members of any paths between the super source and super sink. Any flow occurring on these arcs, at the time represented, does not contribute to the total source to sink flow during the time interval being examined here. For this reason, it is acceptable to remove these arcs from the network representation to simplify the illustration. Figure 5 shows the reduced time-expanded replica which clearly illustrates all temporally repeated network paths that contribute to the source to sink flow during a given interval. The path drawn as a broken line connecting the origin node in the first row to the destination node in the last row is the longest source to sink path from the original sample network. The shortest path, O-c-D, is replicated 5 times in the expanded network illustrating derivation of the sigma parameter for the dynamic models which represents the number of temporal path repetitions.

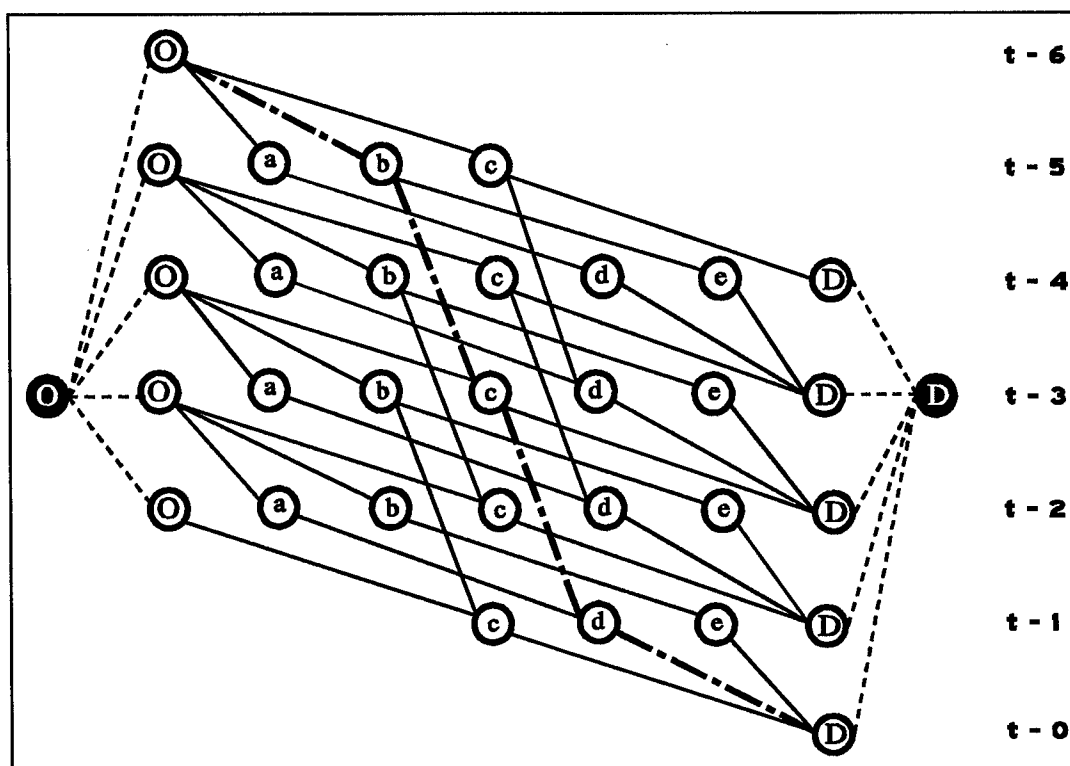


Figure 5: Reduced Time-Expanded Replica of the Sample Network

**5.1.2 Bounding Network Performance.** The performance models proposed in this methodology bound dynamic performance by considering network performance over a time interval as illustrated by the time-expanded replica. Although this is the case, the model formulations account for the dynamic nature of the problem without actually building the expanded network. For this reason, only the parameters of the original network are necessary in the network description file. The network description file constructed to represent this sample network and serve as input to the FORTRAN program is given in Appendix B.

This network description file was used to construct several sets of bounds for the expected performance of the sample network. For the absolute performance measures, throughput and reliability, only one set of bounds are required for a given period. Bounds on the expected relative performance measures, though, vary depending on the utilization level.

To examine the behavior of the relative performance measures, bounds will be obtained at various levels of network utilization ranging from very light flow to flow approaching capacity. The results are summarized in Table 15.

	Case 1	Case 2	Case 3	Case 4
Period	10	10	10	10
Utilization	18	36	54	72
LB Reliability	0.946477	0.946477	0.946477	0.946477
UB Reliability	0.950711	0.950711	0.950711	0.950711
LB Throughput	73.34	73.34	73.34	73.34
UB Throughput	88.65	88.65	88.65	88.65
LB BER	0.277500	0.276875	0.274927	0.273950
UB BER	0.277500	0.277500	0.275616	0.274468
LB Delay	2.000000	2.000000	2.291670	2.46875
UB Delay	2.000000	2.09688	2.397920	2.54844

**Table 15: Sample Network Expected Performance Bounds**

The first observation from the results summarized in Table 15 is that the throughput and reliability bounds remain constant over all values of network utilization. The utilization levels for each case in the table were selected based on approximately 25%, 50%, 75%, and 100% of the lower bound on expected maximum flow. It might appear counter intuitive that the results show that as utilization level increased, the bounds on expected bit error rate decreased. This is explained by the fact that flow is always routed on a shortest path bases and the shortest path in this sample network also happens to be one of the least reliable paths in the network. As the level of source to sink flow rises, the model must use some of the other paths that, while longer, are actually more reliable, hence improved BER.

The behavior of the average delay performance is exactly as would be expected for a shortest route model formulation. Since flow routing is by shortest path, increasing flow can never lead to a decrease in the expected value of average delay. The results in Table 15 support this observation. One final observation regarding the results is the fact that the lower and upper bounds on delay and BER for case 1 are equal. This is explained by examining the routing for this extremely light source to sink flow by shortest route. At this low utilization level, all flow can be expected to be routed along the same path, the shortest one. This path has a reliability of 0.7225 which is where the expected BER of 0.2775 come from. Also, the path is length 2 explaining the expected delay.

**5.1.3 Estimating Performance Data.** This section demonstrates how the same models used to bound expected network performance may be used to estimate sequential observations of vector time-series performance. This is accomplished by taking the models back a step to the deterministic case and then using simulated component state vectors to selectively enable or disable all network components. In this way, each iteration with a new state vector yields a unique, deterministic solution for source to sink flow which is used to compute the various performance measures based on either max flow or min cost flow models. The reliability performance measure still requires too much computational effort to get an exact value so one of the bounds must be used. Figure 6 graphs the estimated network performance over time against the bounds on expected performance.

Each of the four graphs in Figure 6 show the estimated data sets plotted against the bounds computed with the models proposed earlier. The dashed horizontal lines show the upper and lower bounds (only upper bound for reliability) computed for the expected value of each performance measure. The other horizontal lines, shown dotted instead of dashed,



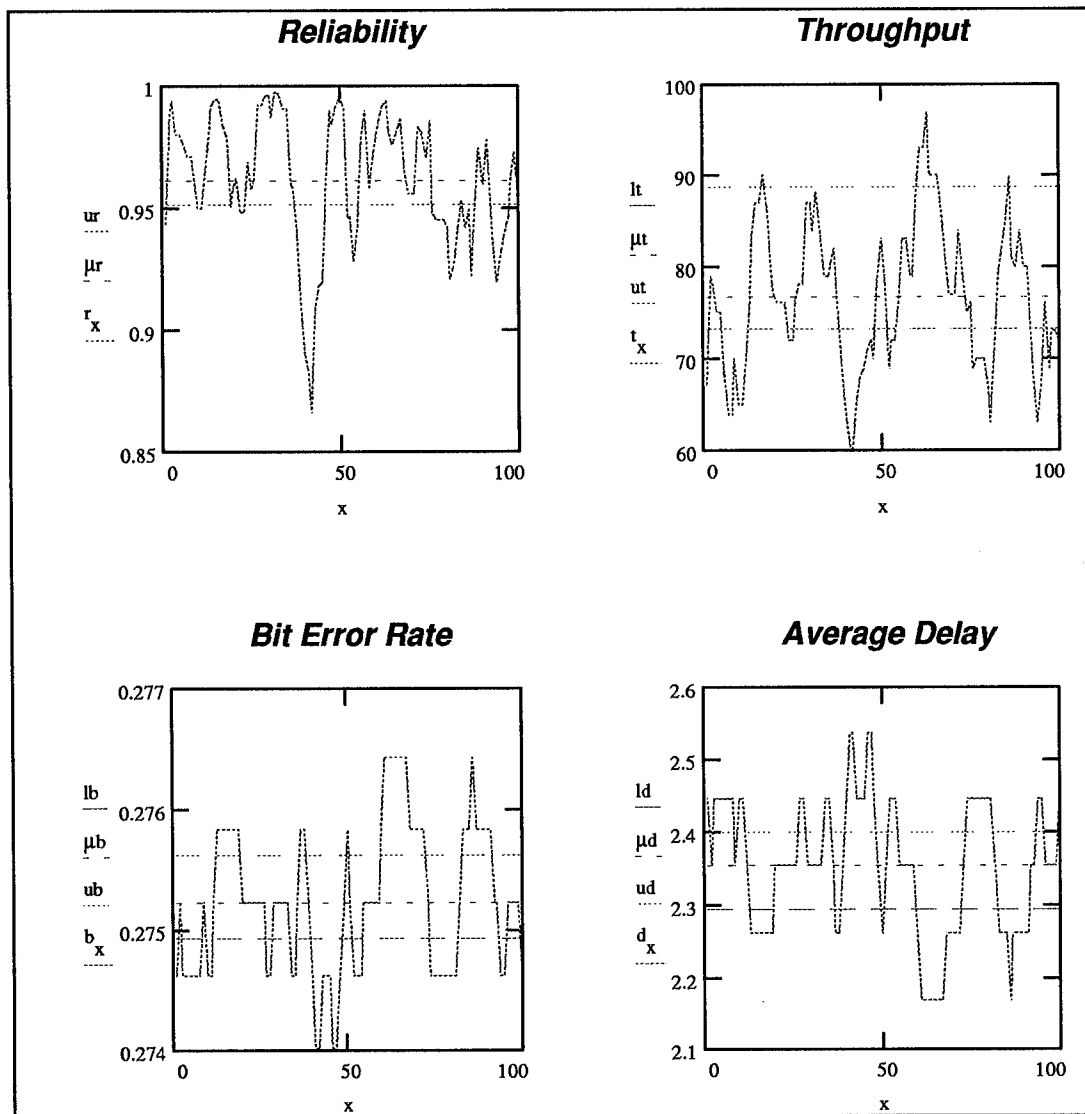


Figure 6: Estimated Vector Time-Series Performance Data

represent the overall sample mean for the 100 observations that were estimated. These graphs show the behavior of each performance measure over time with respect to the bounds obtained for the expected value. For throughput and the two relative performance measures, the sample data mean falls inside of the range defined by the upper and lower bounds. For the reliability measure, the estimated data is observed upper bound on reliability, rather than exact reliability, and the sample mean falls close to the computed upper bound as would be expected.

Another observation from the graphs in Figure 6 is that BER and delay have a strong inverse relationship. This agrees with the same network unique tendency that was exhibited by the bounds on these relative performance measures when utilization level was varied. As previously noted, the shorter source to sink paths in this particular network are also the less reliable paths. If the state of the network is such that the shorter paths may be used, the delay is lower, but the BER suffers. When the longer paths must be used, delay is less appealing, but BER improves. This relationship will vary depending on individual network design.

The models proposed here allow bounds to be placed on the expected value of various network performance measures, but provide no insight into the variance of these measures. An effort was made to derive some functional relationship between the performance measure bounds and the variance of the performance measure (ie. upper bound equal to expected value plus some constant times the standard deviation). This would be extremely useful in the development of control charts for monitoring performance over time operationally, but it was observed that the relationship between the bounds and the underlying performance measure distributions was highly dependent on the specific network topology and parameters.

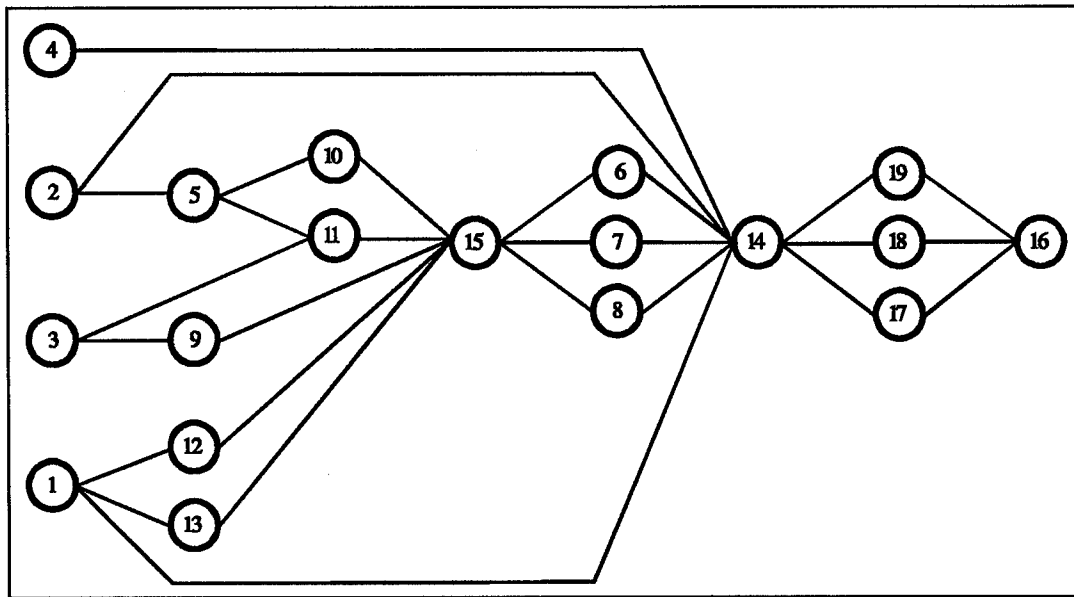
The previously demonstrated capability to estimate vector time-series observations of performance provides an alternate way of using the models for control chart construction. The FORTRAN implementation of the performance models provides an easy way to quickly generate large numbers of observations of any or all of the measures. With a large enough sample, a good estimate of the performance measure variances may be obtained. The expected value bounds and the variance estimate together can be used to construct a control chart for performance monitoring. This sort of chart would be used to evaluate network performance to see if it is within a certain number of standard deviations from the mean.

Control charts like this would normally be constructed by actually observing performance on an operational network and building a large data set from which the mean and variance would be estimated. The use of the models proposed here provides the capability to quickly estimate a large data set for control chart construction without actually monitoring the real system. This is convenient for evaluating a possible change in network topology, change in network utilization level (new network user), or even a change in network monitoring strategy (period of the dynamic models). With this approach to control chart construction, it is relatively easy to evaluate the effects of these possible modifications without actually performing them. Note that it is assumed here that the models have been validated and accurately reflect the actual system.

## **5.2 Case Study**

Now the methodology just demonstrated on a small sample network is applied to three larger, more realistic network designs. The three networks to be analyzed here are taken from the work of Yim[31]. For each network the analysis will be conducted by first presenting a network description and then a summary of the expected performance at various levels of network utilization. Complete results of this analysis along with a listing of the network description files will be given in appendices c, d, and e respectively.

**5.2.1 Network A.** The first network considered here is illustrated in Figure 7. The parameters describing all of the components of this network are given in tables 16 and 17. All of the arcs in Table 17 have been assigned a traversal time of 1 so that the average delay will be measured as a number of 'hops'. This network, as described by the figure and two tables, violates two of the assumptions made previously in Chapter I and used to construct all



**Figure 7: Topology of Network A**

of the performance models. The two violated assumptions are that the network nodes are 100% reliable and that all arcs have a very high reliability parameter (upwards of 0.9). The issue of the unreliable nodes can be handled easy enough by replacing each failing node with two reliable nodes connected by an arc with the reliability parameter of the old node. The problem with the low arc reliability parameters is more serious. This assumption is vital to the reliability models because of the technique that is used for enumerating the most probable

Node	1	2	3	4	5	6	7	8	9	10
Reliability	1.0	0.3	0.7	0.5	0.8	1.0	0.3	0.7	0.5	0.8
Capacity	inf	inf	inf	inf	inf	inf	inf	inf	inf	inf
Node	11	12	13	14	15	16	17	18	18	
Reliability	1.0	0.3	0.7	0.5	0.8	0.8	0.7	0.3	1.0	
Capacity	inf	inf	inf	inf	inf	inf	inf	inf	inf	

**Table 16: Node Parameters For Network A**

Start Node	End node	Reliability	Capacity	Traversal Time
1	12	1.0	1200	1
1	13	1.0	1200	1
1	14	1.0	1200	1
2	5	0.3	1200	1
2	14	0.6	1200	1
3	9	1.0	1200	1
3	11	1.0	1200	1
4	14	1.0	1200	1
5	10	0.6	1200	1
5	11	0.7	1200	1
6	14	0.6	4800	1
7	14	0.6	4800	1
8	14	0.3	4800	1
9	15	1.0	4800	1
10	15	0.6	4800	1
11	15	1.0	4800	1
12	15	0.7	4800	1
13	15	1.0	4800	1
14	17	0.3	4800	1
14	18	0.6	4800	1
14	19	0.6	4800	1
15	6	0.3	4800	1
15	7	0.6	4800	1
15	8	0.7	4800	1
17	16	0.7	4800	1
18	16	0.6	4800	1
19	16	0.3	4800	1

**Table 17: Arc Parameters For Network A**

states. The most probable networks states are assumed to be those with only a small number of failed components. With the extremely low component reliabilities in this network, this assumption no longer holds and the bounds given by the reliability models are not very tight.

After replacing each failing node with two reliable nodes and a failing arc, the network may be analyzed with the Fortran program even though the low component reliabilities render some of the output meaningless. This is done not so much to analyze the network, but to validate the results of the models proposed here against Yim's results. Yim computed the static bounds on the expected throughput for this network. The Fortran program is used here to estimate static network performance in terms of per unit time throughput using the dynamic models.

To do this, a series of maximum expected flow bounds are generated with increasing values for period. Recall that dynamic throughput represents the maximum source to sink flow over a set time interval while static throughput represents maximum source to sink flow per unit time. Dividing dynamic throughput by the value of its period yields a performance measure in the same per unit time format as static throughput. The graphs in Figure 8 illustrate that as the period gets arbitrarily large, dynamic flow per unit time converges on the static value for both the upper and lower bounds.

In these graphs the horizontal line shows the static value for the expected maximum flow bounds found by Yim. The other line represents the behavior of the dynamic per unit time flow estimate as period is increase by ten starting at 10 and ending at 1000. Note that the x-axis in each graph is used for period and is displayed in logarithmic scale. From these graphs it is apparent that as period get arbitrarily large, the dynamic estimate converges with Yim's static value, supporting the correctness of this implementation.

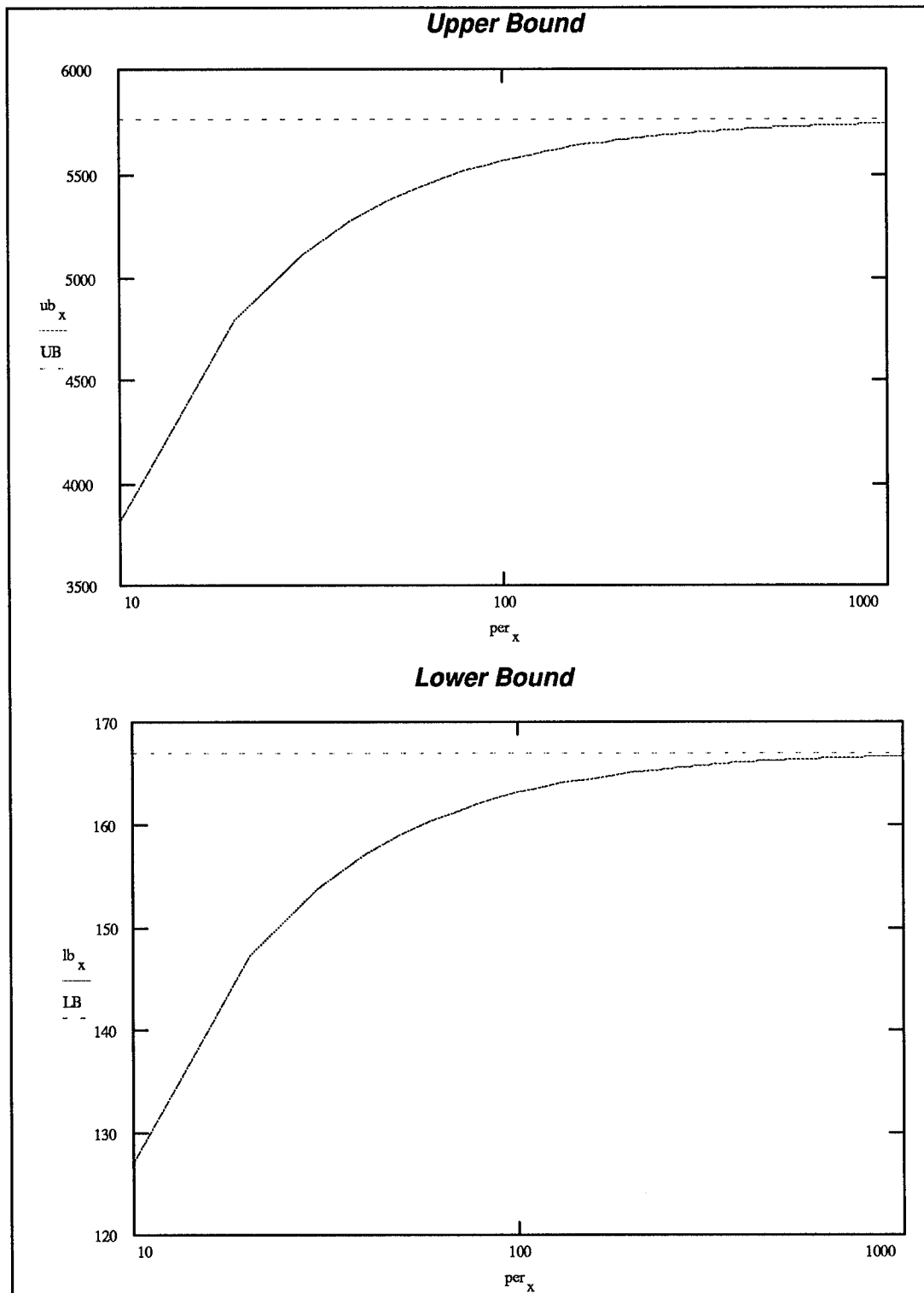


Figure 8: Estimating Static Performance

After verifying that the dynamic models can be used to estimate the static performance measure values, the network is modified to conform more closely to the modelling assumptions. The network node reliabilities are ignored to meet the assumption that only arcs fail and new values are assigned for the network arc reliabilities. The new arc reliability values are assigned arbitrarily between 0.9 and 1.0. These network modifications are incorporated into a new network description file for network a. The results of analyzing this new network are summarized in Table 18.

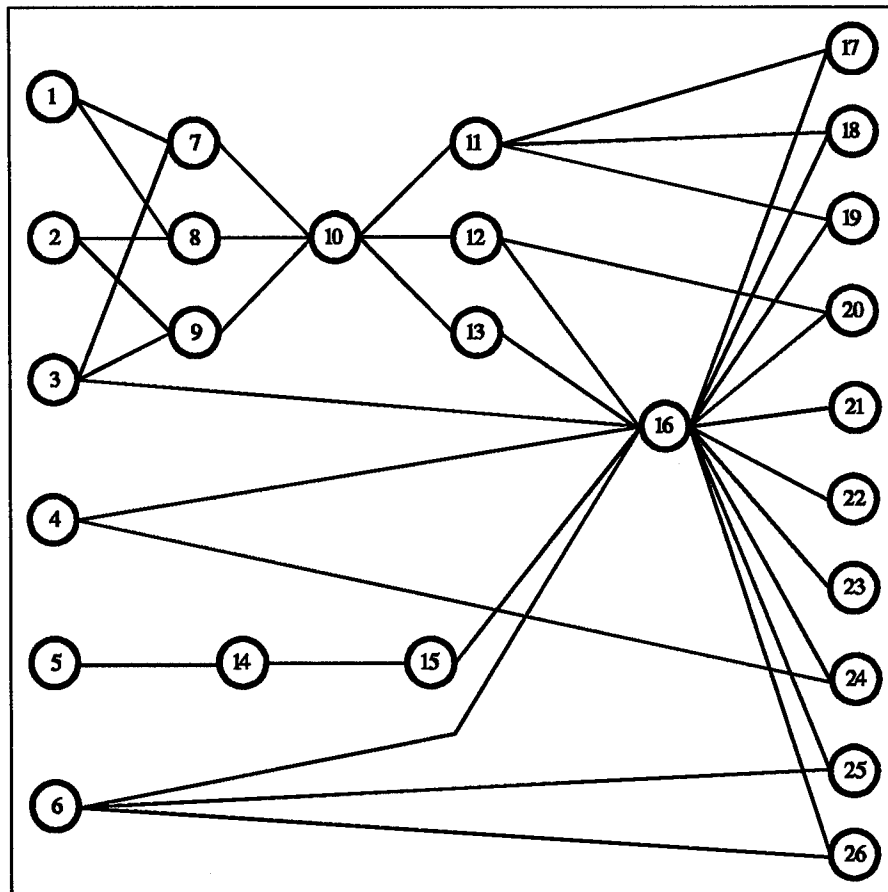
As previously noted, only the relative performance measures, BER and delay, are affected by the changes in utilization level from case to case. For this network though, both relative measures react the same to increases in the level of network utilization. Increasing levels of source to sink flow necessitate using some of the longer paths and in this case, those longer paths are less reliable causing greater BER as well as greater average delay.

	Case 1	Case 2	Case 3	Case 4
Period	10	10	10	10
Utilization	11000	22000	33000	44000
LB Reliability	0.985409	0.985409	0.985409	0.985409
UB Reliability	0.998475	0.998475	0.998475	0.998475
LB Throughput	44733.51	44733.51	44733.51	44733.51
UB Throughput	54774.00	54774.00	54774.00	54774.00
LB BER	0.146017	0.160184	0.181076	0.210971
UB BER	0.183517	0.197087	0.204342	0.220716
LB Delay	3.000000	3.000000	3.539780	4.154840
UB Delay	3.000000	3.000000	3.807860	4.409540

**Table 18: Expected Performance Bounds For Network A**



**5.2.2 Network B.** This network example had the same problems with assumption violations as Network A and these violations were handled similarly. The network is illustrated in Figure 9 and all of the necessary arc parameters are given in Table 19. The results of the network analysis are summarized in Table 20. The reliability bounds for this network configuration differ from the previous examples in that the upper bound is unitary. The method of enumerating the most probable states implemented here is responsible for this outcome. The set of most probable states used here is composed of all network states in which only five or fewer network components have failed. Examining the topology in Figure 9, it is apparent that there exists no set of five or fewer arcs that when failed will completely



**Figure 9: Topology of Network B**

Start Node	End node	Reliability*	Capacity	Traversal Time
1	7	0.8	150	1
1	8	0.8	200	1
2	8	0.5	750	1
2	9	0.5	750	1
3	7	0.8	200	1
3	9	0.5	750	1
3	16	0.6	150	1
4	16	0.8	200	1
4	24	0.5	600	1
5	14	0.8	1200	1
6	16	0.5	1200	1
6	25	0.6	75	1
6	26	0.6	75	1
7	10	0.5	1200	1
8	10	0.7	1200	1
9	10	0.5	2400	1
10	11	0.5	1200	1
10	12	0.7	1200	1
10	13	0.7	1200	1
11	17	0.5	1200	1
11	18	0.5	75	1
11	19	0.5	1200	1
12	16	0.7	1200	1
12	20	0.5	1200	1
13	16	0.7	600	1
14	15	0.8	1200	1
15	16	0.8	1200	1
16	17	0.6	75	1
16	18	0.6	75	1
16	19	0.6	75	1
16	20	0.6	75	1
16	21	0.6	75	1
16	22	0.6	75	1
16	23	0.6	75	1
16	24	0.6	75	1
16	25	0.6	75	1
16	26	0.6	75	1

\*Original reliability values

**Table 19: Arc Parameters For Network B**

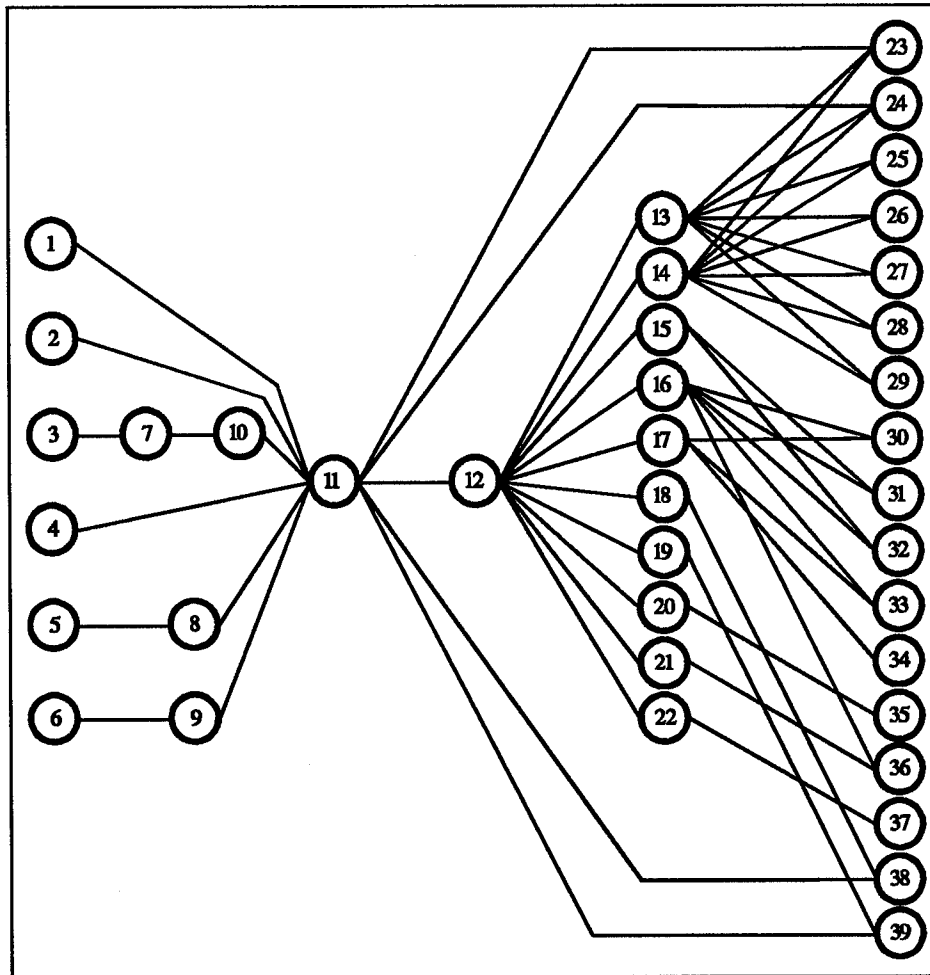
	Case 1	Case 2	Case 3	Case 4
Period	10	10	10	10
Utilization	7500	15000	22500	30000
LB Reliability	0.964520	0.964520	0.964520	0.964520
UB Reliability	1.000000	1.000000	1.000000	1.000000
LB Throughput	26697.26	26697.26	26697.26	26697.26
UB Throughput	28845.90	28845.90	28845.90	28845.90
LB BER	0.047239	0.080083	0.118194	0.140185
UB BER	0.051629	0.097407	0.147769	0.193784
LB Delay	1.048300	1.709710	2.473140	2.741730
UB Delay	1.048300	1.760510	2.507010	2.804290

**Table 20: Expected Performance Bounds For Network B**

cut off the source node from the sink node. Keep in mind that this network configuration has 6 source nodes and 10 sink nodes. In the code, this is handled by the addition of a super source node and a super sink node along with the necessary connecting arcs.

Since there are no cut sets of five or fewer arcs, more than 96% of the total state space has been accounted for, but all of the states considered have been connected states. This accounts for the unitary upper bound on reliability. Although this leads to a meaningless upper bound, the lower bound is still very useful. This situation is expected to arise for any similar network configuration where there are enough parallel source to sink paths to exclude any cut sets of five or fewer arcs.

**5.2.3 Network C.** As for the other two networks, the node reliabilities are removed and the arc reliabilities are replaced in order to conform to all assumptions before any analysis is attempted. Figure 10 shows the network's topology while the network arc parameters are all



**Figure 10: Topology of Network C**

given in Table 21. The results of the network analysis are summarized in Table 22. Here again, there are no small cut sets so only connected states are represented in the set of most probable states. For this reason, only the lower bound on reliability is meaningful in this analysis.

Case 4 of this analysis had a utilization level of 50,000 while the upper bound on the expected maximum flow in this case is less than 44,000. This demonstrates how the Fortran implementation of this methodology handles infeasible levels of source to sink flow. In the

Start Node	End node	Reliability*	Capacity	Traversal Time
1	11	0.9	1200	1
2	11	1.0	1200	1
3	7	1.0	300	1
4	11	0.6	1200	1
5	8	0.3	1200	1
6	9	0.6	1200	1
7	10	1.0	300	1
8	11	0.9	1200	1
9	11	1.0	1200	1
10	11	0.7	300	1
11	12	0.9	9600	1
11	23	0.6	75	1
11	24	0.3	75	1
11	38	0.6	1200	1
11	39	0.7	1200	1
12	13	1.0	4800	1
12	14	1.0	4800	1
12	15	0.6	4800	1
12	16	0.3	4800	1
12	17	0.6	4800	1
12	18	0.7	2400	1
12	19	0.9	4800	1
12	20	1.0	4800	1
12	21	1.0	4800	1
12	22	0.6	2400	1
13	23	0.3	4800	1
13	24	0.6	4800	1
13	25	0.7	2400	1
13	26	0.9	1200	1
13	27	1.0	1200	1
13	28	1.0	1200	1
13	29	0.3	1200	1
14	23	0.6	4800	1
14	24	0.3	4800	1
14	25	0.6	2400	1
14	26	0.7	1200	1
14	27	0.9	1200	1
14	28	1.0	1200	1
14	29	1.0	1200	1
15	31	0.6	2400	1
15	32	0.3	1200	1
16	30	0.6	300	1
16	31	0.7	2400	1
16	32	0.9	1200	1
16	33	1.0	300	1
16	36	1.0	2400	1
17	30	0.6	1200	1
17	33	0.3	300	1
17	34	0.6	300	1
18	38	0.7	1200	1
19	39	0.9	1200	1
20	35	1.0	2400	1
21	36	1.0	2400	1
22	37	0.6	600	1

\*Original reliability values

**Table 21: Arc Parameters For Network C**

	Case 1	Case 2	Case 3	Case 4
Period	10	10	10	10
Utilization	12500	25000	37500	50000
LB Reliability	0.840830	0.840830	0.840830	0.840830
UB Reliability	1.000000	1.000000	1.000000	1.000000
LB Throughput	38944.66	38944.66	38944.66	38944.66
UB Throughput	43453.49	43453.49	43453.49	43453.49
LB BER	0.037837	0.094364	0.149639	0.155960
UB BER	0.100696	0.119064	0.168344	0.191326
LB Delay	2.000000	2.249540	3.061380	3.239500
UB Delay	2.000000	2.296790	3.140740	3.359330

**Table 22: Expected Performance Bounds For Network C**

full program output for this network, given in Appendix E, it is seen that when this particular case was run, the program returned the message, 'network error'. This indicates that the utilization level has been replaced by the maximum feasible flow for each relative performance model. This means that the values for the bounds on the relative performance measures in case 4 are for maximum network utilization as opposed to the utilization level indicated. This presents a means of analyzing 'worst case' network performance.

## ***VI. Validation***

This chapter presents techniques for validating the causal models proposed here against an operational communications network. This is accomplished by comparing a set of estimated vector time-series performance data from the causal models to a set of time-series data from an operational network. The first approach summarizes each data set with a set of estimating equations, comparing the estimating equations for similarities. The other approach analyzes a third set of vector time-series data that is made up of the residuals between the first two. In each case, the goal is to show that the models accurately reflect the real world.

### ***6.1 Obtaining Data Sets***

Ideally, the causal models would be constructed with network topology and component parameters taken from a particular network of interest. The period and utilization level to be examined would correspond to the usage and monitoring policy from the actual network. In this way, the actual and estimated time-series performance data would be comparable and comparisons between the two would be meaningful. This approach to gathering validation data sets was not possible for this effort.

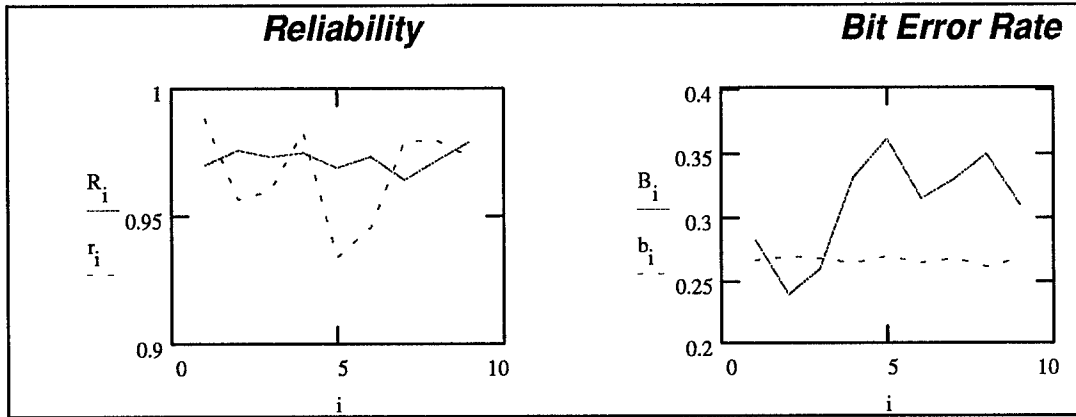
Instead, the 'actual' data set is drawn from monthly executive summary reports provided by the sponsor and the 'estimated' data is based on the first example network presented in the previous chapter. The topology of the network the summary data is drawn from is unknown so an heroic assumption is made that the structure of the actual and model networks are similar and comparing data sets is reasonable. The statement that the networks are similar includes both the topologies and the component parameters.

The next step is to find the intersection between the performance measures proposed in this study and those that were included in the executive summaries. While throughput and average delay are not mentioned in the summary, bit error rate is there along with availability (identical to the reliability measure proposed here). This limits the validation to examining just two performance measures, reliability and bit error rate. Another limitation of the validation effort is that there are only 9 executive summaries available so the study will be limited to comparing two performance data sets of 9 observations each.

An effort must be made to make the estimated data set comparable in time coverage to the observed data set. The arc capacity parameters in the model are all given in BAUD (bits per second) and the units of flow are bits. The time increments are seconds so if the period is 10, the throughput value from the model would give the total of bits that can flow source to sink in ten seconds. This time scale is considerably different from that of the observed data where each observation represents performance over an entire month.

To get the model to provide performance data estimates in the same monthly format as the actual data, model parameters will be altered and data will be aggregated. The first step is to change the bits per second arc capacities to bits per hour by multiplying by a constant. Then, the period is set to 24, providing 'daily' performance estimates which are combined into groups of 30 and averaged. In this way, the causal models can generate vector time-series performance data sets comparable to the actual data taken from the monthly executive summaries. Figure 11 shows the observed performance data set plotted against the estimated performance data. In this figure, reliability and bit error rate are plotted on separate graphs and on each graph the broken line shows the estimated performance data while the solid line is used for the actual data.





**Figure 11: Validation Data Sets**

The graphs of observed versus estimated performance data show that the two data sets are at least in the same range even though time-series behavior seems quite different. For reliability, the estimated data displays considerably more variance while the exact opposite is observed in the BER data. The discrepancies between observed and estimated time-series behavior must be expected considering that the causal models are being used to estimate data from a network for which topology and component parameters are unknown. It follows that the assumptions made here to facilitate comparing these two sets of data may not hold up, and the results of the validation effort to come may not be very accurate, but the validation methodology is sound and is appropriate for analysis under conditions where the assumptions are more reasonable.

## 6.2 Calibrating Estimating Equations

In this section, estimating equations are fit for both sets of performance data and compared for similarities that would suggest that the time-series estimated by the models accurately represents the behavior of the actual data. The estimating equations take the form of simultaneous linear equations capturing both the temporal and multivariate correlation in the

vector time-series data sets. The simultaneous equations are calibrated by the process of two stage least squares as described in chapter 2. The validation data sets along with the complete output from the calibration process are given in appendix f and the calibrated estimating equations are summarized in table 23.

Variables:	
$R_i$	-- Observation 'i' of observed reliability
$B_i$	-- Observation 'i' of observed bit error rate
$r_i$	-- Observation 'i' of estimated reliability
$b_i$	-- Observation 'i' of estimated bit error rate
Equations Derived From Observed Data:	
$R_i = 0.990 \cdot R_{i-1} + 0.035 \cdot B_i$	
$R_{i-1}$ - significant at 99% confidence level	
$B_i$ - significant at 44% confidence level	
$B_i = 0.502 \cdot B_{i-1} + 0.161 \cdot R_i$	
$B_{i-1}$ - significant at 81% confidence level	
$R_i$ - significant at 81% confidence level	
Equations Derived From Estimated Data:	
$r_i = 0.263 \cdot r_{i-1} + 2.675 \cdot b_i$	
$r_{i-1}$ - significant at 56% confidence level	
$b_i$ - significant at 94% confidence level	
$b_i = 0.599 \cdot b_{i-1} + 0.110 \cdot r_i$	
$b_{i-1}$ - significant at 97% confidence level	
$r_i$ - significant at 90% confidence level	

**Table 23: Calibrated Estimating Equations**

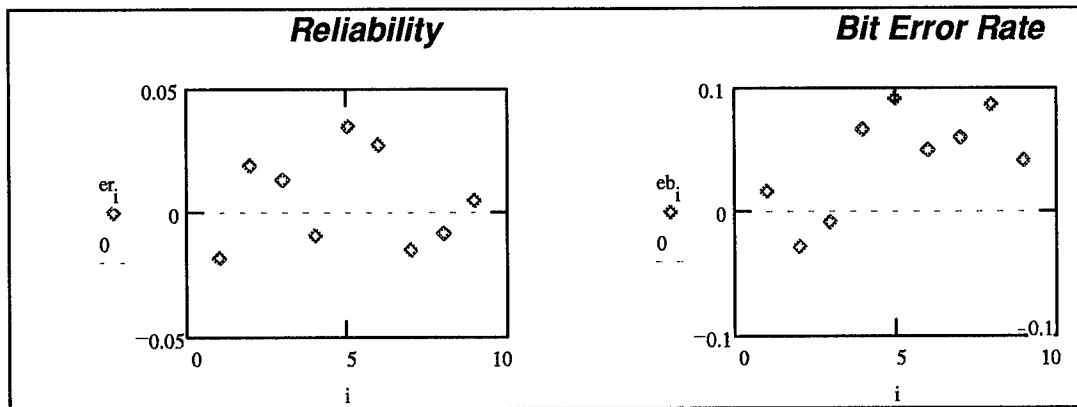
Given two estimating equations for each performance measure, one calibrated from the observed data and one from the estimated data, some observations may be made concerning model validity. These equations summarize the temporal and multivariate correlations of the two data sets and comparing parameters between equations shows significant variation, especially for the reliability estimating equations. For the observed data, only the lagged reliability variable is significant while in the same equation calibrated for the estimated data, only the BER variable is significant. This observation is supported by the original graphs of the data in figure 11. The reliability data graph shows that the reliability values observed in

the operational network did not exhibit much variation explaining why in the estimating equation the previous reliability observation is so significant in estimating the next reliability observation.

Examining plots of the two data sets and comparing parameters between the estimating equations it appears that the two data sets exhibit some significantly different properties. Based on these results it would be hard to support the claim that the models are accurately representing the real system. This is not surprising considering that nothing is known about the network used to obtain the observed data set. If an accurate model were built for a network with known topology and parameters, the graphical and numerical comparisons demonstrated here would be expected to provide evidence supporting model validity.

### ***6.3 Analyzing Time-Series Residuals***

The previous look at validation involved considerable subjective analysis of the properties of two sets of vector time-series performance data. In this section, an attempt is made at outlining a more objective method for model validation involving examination of the residuals between the two data sets. By taking the difference of the two reliability data vectors and the two bit error rate vectors, a pair of error vectors are obtained. If the models are valid then the data sets are from the same population and these error vectors should be essentially white noise. The goal in this more objective validation approach is to show that there is not any significant temporal or multivariate correlation in the vector time-series error data set. If no significant temporal or multivariate correlation is found in the residual data and the data demonstrates homoscedasticity, then the residual data is merely white noise and the models are validated.



**Figure 12: Time-Series Residual Plots**

Before analyzing the residual data for temporal or multivariate correlation an attempt is made at showing homoscedasticity. Figure 12 shows plots of the time-series residuals for both reliability and bit error rate. To show homoscedasticity, the residual plots need to have a mean near zero, a constant variance, and no evidence of a trend or pattern. The residuals for reliability seem to support this, but the BER residuals fall a bit short. In the BER residual plot, observations 4 through 9 are all positive and very further from zero than the first three observations. It appears that the homoscedasticity assumption is violated for the BER data.

Now, the issue of multivariate correlation between the residual vectors will be considered. The objective is to determine if significant correlation exists between the two vectors or between one of the vectors and a lagged representation of the other. Appendix f contains the results of the correlation analysis for the original vectors and for each original vector against first through fourth order lags of the other. It is important to note that the more higher order lags cause loss of degrees of freedom and the data set in question has only nine data points. For this reason, only the original vector correlation and the two first order lag correlations will be considered here. These three correlation terms are all less than 0.2 and do not provide evidence of any significant multivariate correlation.

The issue of temporal correlation is addressed by inspecting the autocorrelations and partial autocorrelations of the individual time-series residual vectors to determine if there is enough temporal correlation to support a significant temporal model. To indicate the form of a potentially significant moving average model, the autocorrelations are examined searching for  $p$  significant positive autocorrelation values before they converge to zero for an  $MA(p)$  model specification. The reliability residual autocorrelations show no significant autocorrelation values, but one significant value is found for the BER residuals. This indicates that the BER residual data could potentially be fit with a significant  $MA(1)$  model. This model specification, after calibration, is significant at an 80% confidence level, but has an  $R^2$  of only 0.25, providing weak evidence at best for the existence of temporal correlation in the residual data. With only nine observations, it is extremely difficult to use the partials to indicate the form of a potential autoregressive model. The pattern to look for would be  $q$  significant non zero partials then convergence to zero for an  $AR(q)$  model.

#### ***6.4 Overall Model Validity***

The findings of the subjective and objective analysis of the time-series residual data are combined to make a judgement on the model validity. This judgement is based on how well the estimated data reflects the real system based on how similar are the properties of the two sets of vector time-series data and how close the residual data comes to white noise. Lacking significant evidence for or against model validity, additional data set could be obtained to build a stronger case one way or the other.

In this particular example, both the subjective and objective analysis provided evidence questioning the validity of the underlying models. In the subjective comparison of

the two data set with graphs and estimating equations, the graphs showed that while in each case, the means were comparable, the variances differed significantly. Also, the estimating equations demonstrated different relationships between the measures from one data set to the other, especially in the case of reliability. In the objective analysis, homoscedasticity was lacking for the BER residuals and there was some evidence for a MA(1) temporal model fit to these residuals.

Each of these separate analysis provided evidence against model validity so either the models are found not valid or further analysis is performed. The validation effort performed here may not be very credible for two reasons. First and most important, nothing is known about the underlying topology and parameters for the network that was the source of the observed data. The network topology and parameters used to generate the estimated data set were chosen from the example networks analyzed previously by selecting the configuration that produced performance data closest to the observed data set. In this way, the estimated data set was similar enough to allow comparisons with the observed data, but the analysis of the three example networks in Chapter V showed just how much network performance is affected by network design. The assumption that the network used to generate the estimated data set has a similar structure to that of the operational network is questionable at best.

The second issue with the credibility of the validation effort here is the small number of observations that were available. It is difficult to make any confident conclusions from a time-series analysis of only nine observations. Both of these issues take away from this particular validation effort, but the validation methodology is sound under more ideal circumstances.

## ***VII. Conclusions and Recommendations***

This chapter provides a summary of the research performed, its conclusions, and recommendations for future research

### ***7.1 Summary***

The goal of this research effort was to identify and model network performance measures that may be useful in the proactive monitoring of stochastic network performance. These performance measures must be modelled in such a way as to be sensitive to the level of flow of the network being examined and whatever monitoring strategy is implemented on this network. To achieve this goal, relative dynamic network performance measures sensitive to utilization level and dynamic period were derived from more traditional absolute performance measures. These new performance measures were demonstrated on three example network structures reflecting realistic topologies and component parameters. Then a validation methodology was presented for use in validating the causal models against the real world system.

From the areas of reliability, availability, and degradation identified by the sponsor as likely candidates for proactive monitoring, average delay and bit error rate were selected as the relative performance measures to be modeled. The development of these models was accomplished by first extending existing throughput and reliability models to a dynamic network flow representation. The maximum dynamic network flow models for these absolute performance measures were then converted to minimum cost dynamic flow for a constant utilization level. Variations of the resulting models were used to generate bounds on the

expected value of the two relative network performance measures average delay and bit error rate for a given network structure and set period and utilization level.

The extended absolute performance measures models along with the new relative performance measure models were demonstrated on three realistic network structures. In this demonstration single bounds were computed for the absolute measures while multiple bounds were found for the relative measures. The multiple bounds were necessary for the relative measures to examine their sensitivity to variations in the utilization level. With this in mind, the bounding models were run with utilization levels selected to demonstrate expected performance at loading ranging from relatively low to the absolute upper bound on source to sink flow.

The final effort was to present a methodology for validating network performance models, these or others, against the operational computer communication network being modelled. This methodology consisted of two distinct areas, both based on comparing an estimated data set with an observed data set attempting to demonstrate that the causal models accurately reflect the behavior of the actual system. The first validation area was a subjective comparison of the data sets based on graphical plots and summarizing estimating equations. The other area was a more objective look at the observed versus estimated residual data attempting to demonstrate a lack of temporal or multivariate correlation between the error vectors.

## ***7.2 Conclusions***

There were several conclusions drawn from this research effort. First, static measures of network performance are not useful in the process of proactively monitoring performance in



an operational computer communication network. The dynamic flow models offer one way of capturing the dynamic nature of performance in an operational network. This formulation also provides the ability to adjust the performance models to accommodate the monitoring strategy in a given network. For example, it is easy to adjust the period in the models to reflect the sampling rate used in the actual network monitoring.

Second, measures of absolute network performance like throughput and reliability, while appropriate for the design of networks, are not as useful to the task of operational network monitoring. An absolute performance measure like throughput is an upper bound on the maximum capacity of a given network. Since an operational network is not expected to always be utilized at its maximum capacity, the relative performance measures are more suited to proactive monitoring. They also provide more flexibility in that period and utilization level can be adjusted to provide bounds for different scenarios.

Third, the key to demonstrating model validity is in using the causal models to generate a set of estimated time-series performance data and then providing evidence that this data accurately reflects the actual performance of the network being modelled. Two important factors are necessary for meaningful analysis of the estimated data versus observed data to be possible. First, the models used for generating the estimated data set must be based on an accurate representation of the operational network in terms of topology and component parameters. Second, it is important to have as many observations in the validation data sets as possible since the number of observations directly affects the accuracy of the findings. An accurate network representation and sufficiently large validation data set makes for high confidence in the findings of the subjective and objective data analysis that is the basis of the model validity judgement.

### **6.3 Recommendations**

The following are recommendations for future research:

First, the findings in this effort can be applied to the sponsor's goal of developing control charts for performance monitoring. The models proposed here are useful in defining bounds on various performance measures based on network utilization and monitoring period. An understanding on expected network performance under various conditions could be useful in developing the control chart methodology for recognizing when network performance is 'out of bounds'.

Second, the models proposed here may prove useful in helping the sponsor define level of service agreements with network resource customers. These models can be applied to study the affects of increased network utilization incurred from new network users. Understanding the expected network performance at the new utilization level will provide guidelines on what is feasible to offer the customer in a binding level of service agreement. Also, the agreement itself can be summarized by values of the performance measures presented here among others.

Third, it would be useful to extend the results of this research to the multiuser, multicommodity case. The scope of this effort was limited to single commodity, single origin-destination pair. While this simplification allowed the focus to concentrate on other complicating issues, it restricted the application of these findings to a very limited subclass of possible network configurations. Extending these results to at least multiuser network configurations would expand the application area of these models to a more realistic class of network configurations.

## *Bibliography*

1. Abraham, J. A. "An Improved Algorithm for Network Reliability," IEEE Transactions on Reliability, R-28: 58-61 (April 1979).
2. Aggarwal, K. K. "Integration of Reliability and Capacity in Performance Measure of a Telecommunication Network," IEEE Transactions on Reliability, R-34: 184-186 (June 1985).
3. Aggarwal, K. K. "A Fast Algorithm for Reliability Evaluation," IEEE Transactions on Reliability, R-24: 83-85 (April 1975).
4. Ahuja, Ravindra K. Network Flows. New Jersey: Prentice-Hall, 1993.
5. Aneja, Y. P. and K. P. K. Nair. "Maximal Expected Flow in a Network Subject to Arc Failure," Networks, 10: 45-57 (1980).
6. Ball, Michael O. "Complexity of Network Reliability Computations," Networks, 10: 153-165 (1980).
7. Ball, Michael O. "Computing Network Reliability," Operations Research, 27: 822-838 (July-August 1979).
8. Brown, D. B. "A Computerized Algorithm for Determining the Reliability of Redundant Configurations," IEEE Transactions on Reliability, R-20: 121-124 (August 1971).
9. Bulka, Dov and Joanne Bechta Dugan. "Network Reliability Bounds Using a 2-Dimensional Reliability Polynomial," IEEE Transactions on Reliability, 43: 39-45 (March 1994).
10. Carey, Malachy and Chris Hendrickson. "Bounds on Expected Performance of Networks with Links Subject to Failure," Networks, 14: 439-456 (1984).
11. Chan, Yupo. Forthcoming Book, Boyd & Fraser, 1994.
12. Evans, J. R. "Maximum Flow in Probabilistic Graphs - The Discrete Case," Networks, 6: 161-183 (1976).
13. Ford, L. R. Jr. and D. R. Fulkerson. Flows in Networks. Princeton: Princeton University Press, 1962.
14. Frantzeskakis, L. F. and W. A. Powell. "Bounding Procedures for Multistage Stochastic Dynamic Networks," Networks, 23: 575-595 (October 1993).

15. Heidtmann, K. D. "Smaller Sums of Disjoint Products by Subproduct Inversion," IEEE Transactions on Reliability, 38: 305-311 (August 1989).
16. Heyman, Daniel P. and Matthew J. Sobel. Stochastic Models in Operations Research: Volume I. New York: McGraw-Hill Book Company, 1982.
17. Jansen, J. Improving Stochastic Communication Network Performance: Reliability vs. Throughput. MS thesis, AFIT/GSO/ENS/91D-10. School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, December 1991 (AD-A243655).
18. Kubat, Peter. "Assessing Throughput and Reliability in Communication and Computer Networks," IEEE Transactions on Reliability, 37: 308-311 (August 1988).
19. Lam, Y. F. and V. O. Li. "An Improved Algorithm for Performance Analysis of Networks with Unreliable Components," IEEE Transactions on Communication, COM-34: 496-497 (May 1986).
20. Li, V. O. K. and J. A. Sylvester. "Performance Analysis of Networks with Unreliable Components," IEEE Transactions on Communications, COM-32: 1105-1110 (October 1984).
21. Locks, M. O. "A Minimizing Algorithm for the Sum of Disjoint Products," IEEE Transactions on Reliability, R-36: 445-453 (October 1987).
22. Onaga K. "Bounds on the Average Terminal Capacity of Probabilistic Nets," IEEE Transactions on Information Theory: 766-768 (September 1968).
23. Ove, Frank and Wolfgang Gaul. "On Reliability in Stochastic Graphs," Networks, 12: 119-126 (1982).
24. Page, L. B. and J. E. Perry. "Reliability of Directed Networks using the Factoring Algorithm," IEEE Transactions on Reliability, 38: 556-562 (December 1989).
25. Powell, Warren B. and Linos F. Frantzeskakis. "Restricted Recourse Strategies for Dynamic Networks with Random Arc Capacities," Transportation Science, 28: 3-23 (February 1994).
26. Provan, S. J. and M. O. Ball. "Computing Network Reliability in Time Polynomial in the Number of Cuts," Operations Research, 32: 516-526 (May-June 1984).
27. Sancho, N. G. F. "On the Maximum Expected Flow in a Network," Journal of the Operations Research Society, 39: 481-485 (May 1988).
28. Shier, D. R. "A New Algorithm for Performance Analysis of Communication Systems," IEEE Transactions on Communications, 36: 516-519 (April 1988).

29. Smith, David R. Digital Transmission Systems. New York: Van Nostrand Reinhold, 1993.
30. Wood, K. R. "Factoring Algorithms for Computing K-Terminal Network Reliability," IEEE Transactions on Reliability, R-35: 269-278 (August 1986).
31. Yim, E. Improving the Survivability of a Stochastic Communication Network. MS thesis, AFIT/GOR/ENS/88D-01. School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, December 1988 (AD-A202872).
32. Yoo, Y. B. and Narsingh Deo. "A Comparison of Algorithms for Terminal-Pair Reliability," IEEE Transactions on Reliability, 37: 210-215 (June 1988).

## ***Appendix A. Fortran Implementation***

### ***A.1 Sample Input***

:::::::::::::

net.dat

:::::::::::::

7

10

1 2 0.9 4 1

1 3 0.9 3 1

1 4 0.85 5 1

2 5 0.9 4 1

3 4 0.95 4 2

3 6 0.9 3 1

4 5 0.95 4 2

4 7 0.85 5 1

5 7 0.9 4 1

6 7 0.9 3 1

## A.2 Input Subroutines

:::::::::::::

input.f

:::::::::::::

```
      subroutine input()
c
      parameter (maxn=50)
      parameter (maxa=10)
      parameter (maxp=100)
      integer nn,na,np,c(maxn,maxn),t(maxn,maxn)
      integer arc(maxn,0:maxa),path(maxp,0:maxn)
      real r(maxn,maxn)
      common/network/nn,na,np,c,t,arc,path,r
c
      integer i,o,d,cap,tim
      real rel
c
      open(unit=20,file='net.dat')
      read(20,'(i2)') nn
      read(20,'(i3)') na
      do 100 i=1,nn
         arc(i,0)=0
100    continue
      do 200 i=1,na
         read(20,10) o,d,rel,cap,tim
10      format(i2,1x,i2,1x,f5.3,1x,i2,1x,i2)
         arc(o,0)=arc(o,0)+1
         arc(o,arc(o,0))=d
         c(o,d)=cap
         t(o,d)=tim
         r(o,d)=rel
200    continue
      close(20)
      return
      end
```

### A.3 Path Enumeration Subroutines

```
.....:
enumpath.f
.....:

      subroutine enumpath()
c
      parameter (maxn=50)
      parameter (maxa=10)
      parameter (maxp=100)
      integer nn,na,np,c(maxn,maxn),t(maxn,maxn)
      integer arc(maxn,0:maxa),path(maxp,0:maxn)
      real r(maxn,maxn)
      common/network/nn,na,np,c,t,arc,path,r
c
      integer i,j,flag,level,done,nodes(maxn),tree(maxn,0:maxa)
      external fathom
c
      done=0
      level=1
      np=0
      do 100 i=1,nn
         tree(i,0)=0
         nodes(i)=0
100    continue
         tree(1,0)=arc(1,0)
         nodes(1)=1
         nodes(arc(1,arc(1,0)))=1
         do 200 j=1,arc(1,0)
            tree(1,j)=arc(1,j)
200    continue
         call fathom(tree,level,nodes)
400    continue
         if(nodes(nn).eq.1) then
            np=np+1
            path(np,0)=level+1
            path(np,1)=1
            do 500 j=2,path(np,0)
               path(np,j)=tree(j-1,tree(j-1,0))
500    continue
            done=1
            level=1
            np=0
            nodes=0
            tree=0
            arc=0
            path=0
            r=0
            common/network/nn,na,np,c,t,arc,path,r
         end if
      end do
      return
      end
```



```

500     continue
endif
if(tree(level,0).ne.0) then
    nodes(tree(level,tree(level,0)))=0
    tree(level,0)=tree(level,0)-1
endif
if(tree(level,0).ne.0) then
    nodes(tree(level,tree(level,0)))=1
    call fathom(tree,level,nodes)
endif
if(tree(level,0).eq.0) then
    flag=0
300     continue
        if(level.ne.1) then
            level=level-1
            nodes(tree(level,tree(level,0)))=0
            tree(level,0)=tree(level,0)-1
        endif
        if(level.eq.1) then
            flag=1
        endif
        if(tree(level,0).ne.0) then
            nodes(tree(level,tree(level,0)))=1
            call fathom(tree,level,nodes)
            flag=1
        endif
        if(flag.eq.0) go to 300
        if(level.eq.1) then
            if(tree(level,0).eq.0) then
                done=1
            endif
        endif
    endif
endif
if(done.eq.0) go to 400
print *, 'Depth First Path Enumeration'
do 600 i=1,np
    print *, ' '
    print 10, 'There are ', path(i,0), ' nodes in path number ', i
10    format(1x,a10,i1,a22,i1)
    print 20, '          ', (path(i,j), j=1, path(i,0))

```

```
20      format(1x,a10,10i2)
600  continue
      print *,' '
      print *,' '
      return
end
```

```
:::::::::::::
fathom.f
:::::::::::::
```

```
      subroutine fathom(tree,level,nodes)
c
      parameter (maxn=50)
      parameter (maxa=10)
      parameter (maxp=100)
      integer tree(maxn,0:maxa),level,nodes(maxn)
      integer nn,na,np,c(maxn,maxn),t(maxn,maxn)
      integer arc(maxn,0:maxa),path(maxp,0:maxn)
      real r(maxn,maxn)
      common/network/nn,na,np,c,t,arc,path,r
c
      integer i,j,flag
c
200  continue
      flag=0
      i=tree(level,tree(level,0))
      if(arc(i,0).ne.0) then
        level=level+1
        do 300 j=1,arc(i,0)
          if(nodes(arc(i,j)).eq.0) then
            tree(level,0)=tree(level,0)+1
            tree(level,tree(level,0))=arc(i,j)
            flag=1
          endif
300  continue
          if(tree(level,0).ne.0) then
            nodes(tree(level,tree(level,0)))=1
          endif
        endif
        if(nodes(nn).eq.1) then
          flag=0
        endif
      if(flag.eq.1) go to 200
      return
      end
```

#### A.4 Model Construction Subroutines

::::::::::::

models.f

::::::::::::

```
      subroutine models(p,f)
c
      parameter (maxn=50)
      parameter (maxa=10)
      parameter (maxp=100)
      integer nn,na,np,c(maxn,maxn),t(maxn,maxn)
      integer arc(maxn,0:maxa),path(maxp,0:maxn)
      real r(maxn,maxn)
      integer att(maxn*maxa),pcomp(maxp,0:maxn)
      integer m,n,lda,ir(maxn*maxa),cs(maxp),co(maxp)
      real a(maxn*maxa,maxp),b(maxn*maxa),rb(maxn*maxa),cr(maxp)
      real xl(maxp),xu(maxp),obj,xs(maxp),ds(maxn*maxa)
      common/network/nn,na,np,c,t,arc,path,r
      common/model/m,n,a,lda,b,rb,co,cs,cr,ir,xl,xu,obj,xs,ds,att,pcomp
c
      integer anum(maxn,maxn),p
      real f
c
      m=na
      n=np
      lda=maxn*maxa
      do 100 i=1,m
         ir(i)=1
100    continue
      ir(m+1)=0
      k=0
      do 300 i=1,m
         if(arc(i,0).ne.0) then
            do 400 j=1,arc(i,0)
               k=k+1
               anum(i,arc(i,j))=k
               att(k)=t(i,arc(i,j))
               b(k)=real(c(i,arc(i,j)))
               rb(k)=r(i,arc(i,j))
400          continue
```

```

endif
300 continue
b(m+1)=f*real(p)
rb(m+1)=1.0
do 200 i=1,n
  cs(i)=0
  cr(i)=1.0
  xl(i)=0.0
  xu(i)=999999999.0
  pcomp(i,0)=path(i,0)-1
  do 600 j=2,path(i,0)
    pcomp(i,j-1)=anum(path(i,j-1),path(i,j))
    cs(i)=cs(i)+t(path(i,j-1),path(i,j))
    cr(i)=cr(i)*r(path(i,j-1),path(i,j))
600 continue
    co(i)=p+1-cs(i)
200 continue
do 101 i=1,m
  do 101 j=1,n
    a(i,j)=0.0
101 continue
do 500 i=1,n
  do 500 j=2,path(i,0)
    a(anum(path(i,j-1),path(i,j)),i)=1.0
500 continue
m=m+1
do 505 i=1,n
  a(m,i)=real(co(i))
505 continue
return
end

```

## A.5 State Vector Generation Subroutines

```

:
:
states.f
:
:
      subroutine states(runs,p,nfp)
c
      parameter (maxn=50)
      parameter (maxa=10)
      parameter (maxr=150)
      parameter (maxp=100)
      integer nn,na,np,c(maxn,maxn),t(maxn,maxn)
      integer arc(maxn,0:maxa),path(maxp,0:maxn)
      real r(maxn,maxn)
      integer att(maxn*maxa),pcomp(maxp,0:maxn)
      integer m,n,lda,ir(maxn*maxa),cs(maxp),co(maxp)
      real a(maxn*maxa,maxp),b(maxn*maxa),rb(maxn*maxa),cr(maxp)
      real xl(maxp),xu(maxp),obj,xs(maxp),ds(maxn*maxa)
      common/network/nn,na,np,c,t,arc,path,r
      common/model/m,n,a,lda,b,rb,co,cs,cr,ir,xl,xu,obj,xs,ds,att,pcomp
c
      integer runs,p,max,nnas,ast(maxr,maxn*maxa),nnps,l,pst(maxr,maxp)
      integer nfp(maxr,maxp)
      external rnunf
c
      max=0
      do 100 i=1,np
        if(cs(i).gt.max) then
          max=cs(i)
        endif
100    continue
      nnas=runs+p+max
      do 200 i=1,nnas
        do 200 j=1,na
          ast(i,j)=int(1.0-rnunf()+rb(j))
200    continue
      nnps=runs+p
      do 300 i=1,nnps
        do 300 j=1,np
          l=i

```

```

        pst(i,j)=1
        do 300 k=1,pcomp(j,0)
            pst(i,j)=pst(i,j)*ast(l,pcomp(j,k))
            l=l+att(pcomp(j,k))
300    continue
        do 400 i=1,runs
            do 400 j=1,np
                nfp(i,j)=co(j)
                do 400 k=0,co(j)-1
                    l=i+k
                    nfp(i,j)=nfp(i,j)-pst(l,j)
400    continue
        return
    end

```

## A.6 Model Solution Subroutines

```
.....:
reli.f
.....:
      subroutine reli(pathstat,obs)
c
      parameter (maxr=150)
      parameter (maxn=50)
      parameter (maxa=10)
      parameter (maxp=100)
      integer att(maxn*maxa),pcomp(maxp,0:maxn)
      integer m,n,lda,ir(maxn*maxa),cs(maxp),co(maxp)
      real a(maxn*maxa,maxp),b(maxn*maxa),rb(maxn*maxa),cr(maxp)
      real xl(maxp),xu(maxp),obj,xs(maxp),ds(maxn*maxa)
      real rel(2,maxr),tp(2,maxr),ber(2,maxr),del(2,maxr)
      common/perfdata/rel,tp,ber,del
      common/model/m,n,a,lda,b,rb,co,cs,cr,ir,xl,xu,obj,xs,ds,att,pcomp
c
      integer pathstat(maxp),obs,arcs,state(maxn*maxa)
      integer cou1(maxn*maxa),cou2(maxn*maxa),s,o
      double precision prob,sum,lb,ub
      real arcrel(maxn*maxa),adjust(maxn*maxa)
      external iset,eval
c
      o=abs(obs)
      arcs=m-1
      call iset(arcs,0,cou1,1)
      call iset(arcs,0,cou2,1)
      do 10 i=1,n
         do 10 j=1,pcomp(i,0)
            k=pcomp(i,j)
            if(obs.lt.0)then
               cou1(k)=cou1(k)+co(i)-pathstat(i)
               cou2(k)=cou2(k)+co(i)
            else
               cou1(k)=cou1(k)+int(rb(i)*1000*co(i))
               cou2(k)=cou2(k)+(1000*co(i))
            endif
         10 continue
```



```

do 20 i=1,arcs
  adjust(i)=real(cou1(i))/real(cou2(i))
  if(obs.lt.0) then
    arcrel(i)=adjust(i)
  else
    arcrel(i)=rb(i)*adjust(i)
  endif
20  continue
  lb=0.0
  ub=1.0
  sum=1.0
  do 50 nn=1,arcs
    sum=sum*dbble(arcrel(nn))
50  continue
  lb=lb+sum
  do 100 i=1,arcs
    prob=1.0
    do 150 nn=1,arcs
      if(nn.ne.i) then
        prob=prob*dbble(arcrel(nn))
      else
        prob=prob*dbble(1.0-arcrel(nn))
      endif
150  continue
    sum=sum+prob
    call iset(arcs,1,state,1)
    state(i)=0
    call eval(state,s)
    lb=lb+(dbble(prob)*dbble(s))
    ub=ub-(dbble(prob)*dbble(1-s))
100  continue
  do 200 i=1,arcs-1
  do 200 j=i+1,arcs
    prob=1.0
    do 250 nn=1,arcs
      if((nn.ne.i).and.(nn.ne.j)) then
        prob=prob*dbble(arcrel(nn))
      else
        prob=prob*dbble(1.0-arcrel(nn))
      endif

```

```

250    continue
      sum=sum+prob
      call iset(arcs,1,state,1)
      state(i)=0
      state(j)=0
      call eval(state,s)
      lb=lb+(dble(prob)*dble(s))
      ub=ub-(dble(prob)*dble(1-s))
200  continue
      do 300 i=1,arcs-2
      do 300 j=i+1,arcs-1
      do 300 k=j+1,arcs
        prob=1.0
        do 350 nn=1,arcs
          if((nn.ne.i).and.(nn.ne.j).and.(nn.ne.k)) then
            prob=prob*dble(arcrel(nn))
          else
            prob=prob*dble(1.0-arcrel(nn))
          endif
350  continue
      sum=sum+prob
      call iset(arcs,1,state,1)
      state(i)=0
      state(j)=0
      state(k)=0
      call eval(state,s)
      lb=lb+(dble(prob)*dble(s))
      ub=ub-(dble(prob)*dble(1-s))
300  continue
      do 400 i=1,arcs-3
      do 400 j=i+1,arcs-2
      do 400 k=j+1,arcs-1
      do 400 l=k+1,arcs
        prob=1.0
        do 450 nn=1,arcs
          if((nn.ne.i).and.(nn.ne.j).and.(nn.ne.k).and.(nn.ne.l)) then
            prob=prob*dble(arcrel(nn))
          else
            prob=prob*dble(1.0-arcrel(nn))
          endif

```

```

450    continue
      sum=sum+prob
      call iset(arcs,1,state,1)
      state(i)=0
      state(j)=0
      state(k)=0
      state(l)=0
      call eval(state,s)
      lb=lb+(dble(prob)*dble(s))
      ub=ub-(dble(prob)*dble(1-s))
400    continue
      do 500 i=1,arcs-4
      do 500 j=i+1,arcs-3
      do 500 k=j+1,arcs-2
      do 500 l=k+1,arcs-1
      do 500 mm=l+1,arcs
        prob=1.0
        do 550 nn=1,arcs
          if((nn.ne.i).and.(nn.ne.j).and.(nn.ne.k)
+          .and.(nn.ne.l).and.(nn.ne.mm)) then
            prob=prob*dble(arcrel(nn))
          else
            prob=prob*dble(1.0-arcrel(nn))
          endif
550    continue
      sum=sum+prob
      call iset(arcs,1,state,1)
      state(i)=0
      state(j)=0
      state(k)=0
      state(l)=0
      state(mm)=0
      call eval(state,s)
      lb=lb+(dble(prob)*dble(s))
      ub=ub-(dble(prob)*dble(1-s))
500    continue
      rel(1,o)=lb
      rel(2,o)=ub
      return
      end

```

```

:.....:
eval.f
:.....:

```

```

      subroutine eval(state,s)
c
      parameter (maxr=150)
      parameter (maxn=50)
      parameter (maxa=10)
      parameter (maxp=100)
      integer att(maxn*maxa),pcomp(maxp,0:maxn)
      integer m,n,lda,ir(maxn*maxa),cs(maxp),co(maxp)
      real a(maxn*maxa,maxp),b(maxn*maxa),rb(maxn*maxa),cr(maxp)
      real xl(maxp),xu(maxp),obj,xs(maxp),ds(maxn*maxa)
      real rel(2,maxr),tp(2,maxr),ber(2,maxr),del(2,maxr)
      common/perfdata/rel,tp,ber,del
      common/model/m,n,a,lda,b,rb,co,cs,cr,ir,xl,xu,obj,xs,ds,att,pcomp
c
      integer state(maxn*maxa),s,test
c
      s=0
      do 100 i=1,n
        test=1
        do 150 j=1,pcomp(i,0)
          k=pcomp(i,j)
          test=test*state(k)
150      continue
        if(test.eq.1) then
          s=1
        endif
100    continue
      return
      end

```

```

:.....:
maxflow.f
:.....:

```

```

      subroutine maxflow(pathstat,obs)
c
      parameter (maxr=150)
      parameter (maxn=50)
      parameter (maxa=10)
      parameter (maxp=100)
      integer att(maxn*maxa),pcomp(maxp,0:maxn)
      integer m,n,lda,ir(maxn*maxa),cs(maxp),co(maxp)
      real a(maxn*maxa,maxp),b(maxn*maxa),rb(maxn*maxa),cr(maxp)
      real xl(maxp),xu(maxp),obj,xs(maxp),ds(maxn*maxa)
      real rel(2,maxr),tp(2,maxr),ber(2,maxr),del(2,maxr)
      common/perfdata/rel,tp,ber,del
      common/model/m,n,a,lda,b,rb,co,cs,cr,ir,xl,xu,obj,xs,ds,att,pcomp
c
      integer pathstat(maxp),obs,o
      real cost(maxp),eb(maxn*maxa)
c
      o=abs(obs)
      do 100 i=1,n
        if(obs.lt.0) then
          cost(i)=-1.0*(real(co(i))-real(pathstat(i)))
        else
          cost(i)=-1.0*(real(co(i))-real(pathstat(i)))*cr(i)
        endif
100    continue
      call dlprs(m-1,n,a,lda,b,b,cost,ir,xl,xu,obj,xs,ds)
      tp(1,o)=-1.0*obj
      do 200 i=1,n
        cost(i)=-1.0*(real(co(i))-real(pathstat(i)))
200    continue
      do 300 i=1,m-1
        if(obs.lt.0) then
          eb(i)=b(i)
        else
          eb(i)=b(i)*rb(i)
        endif
      do 300 i=1,m-1

```

```
300  continue
      call dlprs(m-1,n,a,lda,eb,eb,cost,ir,xl,xu,obj,xs,ds)
      tp(2,o)=-1.0*obj
      return
      end
```

:::::::::::::

relative.f

:::::::::::::

```
      subroutine relative(pathstat,obs)
c
      parameter (maxr=150)
      parameter (maxn=50)
      parameter (maxa=10)
      parameter (maxp=100)
      integer att(maxn*maxa),pcomp(maxp,0:maxn)
      integer m,n,lda,ir(maxn*maxa),cs(maxp),co(maxp)
      real a(maxn*maxa,maxp),b(maxn*maxa),rb(maxn*maxa),cr(maxp)
      real xl(maxp),xu(maxp),obj,xs(maxp),ds(maxn*maxa)
      real rel(2,maxr),tp(2,maxr),ber(2,maxr),del(2,maxr)
      common/perfdata/rel,tp,ber,del
      common/model/m,n,a,lda,b,rb,co,cs,cr,ir,xl,xu,obj,xs,ds,att,pcomp
c
      integer pathstat(maxp),obs,o
      real cost(maxp),eb(maxn*maxa),aa(maxn*maxa,maxp),sl(maxp),su(maxp)
      real el(maxp),eu(maxp),tl(maxp),tu(maxp),lsum1,lsum2,usum1,usum2
c
      o=abs(obs)
      do 100 i=1,n
        if(obs.lt.0) then
          cost(i)=(real(co(i))-real(pathstat(i)))*cs(i)
        else
          cost(i)=(real(co(i))-real(pathstat(i)))*cr(i)*cs(i)
        endif
100    continue
      do 200 i=1,m
        eb(i)=b(i)
        do 200 j=1,n
          aa(i,j)=a(i,j)
200    continue
      if(b(m).gt.tp(1,o)) then
        eb(m)=tp(1,o)
        print *,'Network failure'
      endif
      do 300 j=1,n
        if(obs.lt.0) then
```

```

        aa(m,j)=(real(co(j))-real(pathstat(j)))
    else
        aa(m,j)=(real(co(j))-real(pathstat(j)))*cr(j)
    endif
300  continue
    call dlprs(m,n,aa,lda,eb,eb,cost,ir,xl,xu,obj,xs,ds)
    do 400 i=1,n
        if(obs.lt.0) then
            sl(i)=xs(i)*(real(co(i))-real(pathstat(i)))
        else
            sl(i)=xs(i)*cr(i)*(real(co(i))-real(pathstat(i)))
        endif
400  continue
        do 500 i=1,n
            cost(i)=(real(co(i))-real(pathstat(i)))*cs(i)
            aa(m,i)=(real(co(i))-real(pathstat(i)))
500  continue
            do 600 i=1,m
                if(obs.lt.0) then
                    eb(i)=b(i)
                else
                    eb(i)=b(i)*rb(i)
                endif
600  continue
                if(b(m).gt.tp(2,o)) then
                    eb(m)=tp(2,o)
                    print *, 'Network failure'
                endif
                call dlprs(m,n,aa,lda,eb,eb,cost,ir,xl,xu,obj,xs,ds)
                do 700 i=1,n
                    su(i)=xs(i)*(real(co(i))-real(pathstat(i)))
700  continue
                    lsum1=0.0
                    lsum2=0.0
                    usum1=0.0
                    usum2=0.0
                    do 800 i=1,n
                        tl(i)=sl(i)/cr(i)
                        tu(i)=su(i)/cr(i)
                        el(i)=tl(i)-sl(i)

```



```

        eu(i)=tu(i)-su(i)
        lsum1=lsum1+el(i)
        lsum2=lsum2+t1(i)
        usum1=usum1+eu(i)
        usum2=usum2+tu(i)
800  continue
        ber(1,o)=usum1/usum2
        ber(2,o)=lsum1/lsum2
        if((usum1/usum2).gt.(lsum1/lsum2)) then
            ber(1,o)=lsum1/lsum2
            ber(2,o)=usum1/usum2
        endif
        lsum1=0.0
        lsum2=0.0
        usum1=0.0
        usum2=0.0
        do 900 i=1,n
            lsum1=lsum1+(sl(i)*cs(i))
            lsum2=lsum2+sl(i)
            usum1=usum1+(su(i)*cs(i))
            usum2=usum2+su(i)
900  continue
        del(1,o)=usum1/usum2
        del(2,o)=lsum1/lsum2
        if((usum1/usum2).gt.(lsum1/lsum2)) then
            del(1,o)=lsum1/lsum2
            del(2,o)=usum1/usum2
        endif
        return
    end

```

## A.7 Main Program and Makefile

.....

main.f

.....

```
parameter (size=25000)
parameter (maxr=150)
parameter (maxn=100)
parameter (maxa=10)
parameter (maxp=500)
integer nn,na,np,c(maxn,maxn),t(maxn,maxn)
integer arc(maxn,0:maxa),path(maxp,0:maxn)
real r(maxn,maxn)
real rwksp(size)
integer att(maxn*maxa),pcomp(maxp,0:maxn)
integer m,n,lda,ir(maxn*maxa),cs(maxp),co(maxp)
real a(maxn*maxa,maxp),b(maxn*maxa),rb(maxn*maxa),cr(maxp)
real xl(maxp),xu(maxp),obj,xs(maxp),ds(maxn*maxa)
real rel(2,maxr),tp(2,maxr),ber(2,maxr),del(2,maxr)
common/perfdata/rel,tp,ber,del
common/model/m,n,a,lda,b,rb,co,cs,cr,ir,xl,xu,obj,xs,ds,att,pcomp
common/network/nn,na,np,c,t,arc,path,r
common/worksp/rwksp

c
integer p,runs,nfp(maxr,maxp),pathstat(maxp),obs,o,minp,choice,per
integer ast(maxr,maxn*maxa),flag
real f,util,hold
character ans
external input,enumpath,iwkin,models,states,maxflow,relative,reli
external sset,iset

c
call iwkin(size)
call input()
call enumpath()
p=100
f=1.0
call models(p,f)
minp=1
do 50 i=1,np
```

```

        if(cs(i).gt.minp) then
            minp=cs(i)
        endif
50    continue
    choice=1
    print *,' '
75    continue
    print *,' '
    print *,'OPTIONS:'
    print *,'      1) Obtain dynamic performance bounds'
    print *,'      2) Investigate steady state performance bounds'
    print *,'      3) Estimate vector time-series performance data'
    print *,'      4) Exit this program'
    print *,' '
    print *,'Enter 1, 2, 3, or 4 to choose an option. '
    read (*,'(i1)') choice
    if(choice.eq.1) then
        flag=0
        p=100*minp
        call models(p,f)
        call iset(np,0,pathstat,1)
        obs=1
        call maxflow(pathstat,obs)
        print *,' '
        print *,'      GUIDELINES FOR SETTING ',
+         'PERIOD AND UTILIZATION LEVEL'
        print *,' '
        print *,'All of the dynamic models will ',
+         'require a period to be set.'
        print *,'This period should be greater than the longest path.'
        print *,'The lenght of the longest path is ',minp
        print *,' '
        print *,'The relative performance ',
+         'models call for a utilization level.'
        print *,'This is a constant value for the level of s-t flow.'
        print *,'To avoid problems, this value ',
+         'should be kept well under the'
        print *,'      the lower bound on the max expected flow.'
        print *,'Using an arbitrarily large ',
+         'period, ',p,', the lower bound on'

```

```

hold=tp(1,1)/real(p)
print *, '      expected per unit time flow is ',hold
print *, 'Utilization level should be ',
+      'set with respect to period.'
print *, 'This value should be period ',
+      'times some level of flow less than'
print *, '      the stated lower bound on expected max flow.'
print *, 'The following would be a valid parameter selection:'
print *, ' '
print *, 'PERIOD = ',minp*2
print *, 'UTILIZATION =',minp*2,' * ',int(tp(1,1)/real(p*2)),
+      ' = ',(minp*2)*int(tp(1,1)/real(p*2))
print *, ' '
print *, 'Now set parameters according to the guidelines.'
25 continue
print *, 'Enter a value for the dynamic period.'
print *, '      (must be greater than ',minp,')'
read *,per
print *, 'Enter a value for the utilization level.'
print *, '      (must be less than',
+      real(per)*hold,')'
read *,util
util=util/real(per)
print *, ' '
call models(per,util)
call iset(np,0,pathstat,1)
obs=1
call maxflow(pathstat,obs)
call relative(pathstat,obs)
if(flag.eq.0) call reli(pathstat,obs,ast)
flag=1
print *, 'Below are the bounds on expected network performance'
print *, '      when period is set to ',per
print *, '      and per time period flow is ',util
print *, ' '
print *, '      Reliability      ', '      Throughput      ',
+      '      Bit Error Rate      ', '      Average Delay      '
print *, ' ----- ', ' ----- ',
+      ' ----- ', ' ----- '
print 10,rel(1,obs),rel(2,obs),tp(1,obs),tp(2,obs),

```

```

+          ber(1,obs),ber(2,obs),del(1,obs),del(2,obs)
10  format(2f9.6,2f9.2,2f9.6,2f9.5)
    print *, ' '
    print *, 'Do you want to compute bounds for other parameters?'
    print *, '      (Enter y or n)'
    read *,ans
    print *, ' '
    if((ans.eq.'y').or.(ans.eq.'Y')) go to 25
endif
if(choice.eq.2) then
    print *, ' '
    print *, 'For this application utilization level will be'
    print *, ' assumed to be at the lower bound for max flow.'
    print *, ' '
    print *, ' Reliability      ', ' Throughput      ',
+          ' Bit Error Rate ', ' Average Delay  '
    print *, ' ----- ', ' ----- ',
+          ' ----- ', ' ----- '
    call iset(np,0,pathstat,1)
    do 125 obs=1,100
        p=(int(minp/10)+obs)*10
        f=1.0
        call models(p,f)
        call maxflow(pathstat,obs)
        f=tp(1,obs)*0.99/real(p)
        call models(p,f)
        call maxflow(pathstat,obs)
        call relative(pathstat,obs)
        call reli(pathstat,obs,ast)
        print 10,rel(1,obs),rel(2,obs),tp(1,obs)/p,tp(2,obs)/p,
+          ber(1,obs),ber(2,obs),del(1,obs),del(2,obs)
125  continue
    print *, ' '
    print *, 'The steady state per time unit performance estimates'
    print *, ' at an arbitrarily large period ',p,' are below.'
    print *, ' '
    print *, ' Reliability      ', ' Throughput      ',
+          ' Bit Error Rate ', ' Average Delay  '
    print *, ' ----- ', ' ----- ',
+          ' ----- ', ' ----- '

```

```

obs=100
print 10,rel(1,obs),rel(2,obs),tp(1,obs)/p,tp(2,obs)/p,
+      ber(1,obs),ber(2,obs),del(1,obs),del(2,obs)
print *,' '
endif
if(choice.eq.3) then
  print *,' '
  print *,'How many vector time-series to estimate?'
  print *,' (100 observations max)'
  print *,' '
  read *,runs
  p=100*minp
  call models(p,f)
  call iset(np,0,pathstat,1)
  obs=1
  call maxflow(pathstat,obs)
  print *,' '
  print *,'      GUIDELINES FOR SETTING ',
+      'PERIOD AND UTILIZATION LEVEL'
  print *,' '
  print *,'All of the dynamic models will ',
+      'require a period to be set.'
  print *,'This period should be greater than the longest path.'
  print *,'The lenght of the longest path is ',minp
  print *,' '
  print *,'The relative performance ',
+      'models call for a utilization level.'
  print *,'This is a constant value for the level of s-t flow.'
  print *,'To avoid problems, this value ',
+      'should be kept well under the'
  print *,'      the lower bound on the max expected flow.'
  print *,'Using an arbitrarily large ',
+      'period, ',p,', the lower bound on'
  hold=tp(1,1)/real(p)
  print *,'      expected per unit time flow is ',hold
  print *,'Utilization level should be ',
+      'set with respect to period.'
  print *,'This value should be period ',
+      'times some level of flow less than'
  print *,'      the stated lower bound on expected max flow.'

```

```

print *, 'The following would be a valid parameter selection:'
print *, ' '
print *, 'PERIOD = ', minp*2
print *, 'UTILIZATION = ', minp*2, ' * ', int(tp(1,1)/real(p*2)),
+      ' = ', (minp*2)*int(tp(1,1)/real(p*2))
print *, ' '
print *, 'Now set parameters according to the guidelines.'
print *, 'Enter a value for the dynamic period.'
print *, '      (must be greater than ', minp, ') '
read *, per
print *, 'Enter a value for the utilization level.'
print *, '      (must be less than ',
+      real(per)*hold, ') '
read *, util
util=util/real(per)
print *, ' '
call models(per, util)
call states(runs, p, nfp, ast)
print *, 'Below is the estimated vector time-series data.'
print *, ' '
print *, '  Rel      MaxFlow      BER      DEL      '
print *, '-----'
12  format(f8.6, f10.2, 1x, f9.6, f10.5)
do 100 obs=1, runs
  o=-1*obs
  do 200 j=1, np
    pathstat(j)=nfp(obs, j)
200  continue
    call maxflow(pathstat, o)
    call relative(pathstat, o)
    call reli(pathstat, o, ast)
    print 12, rel(2, obs), tp(2, obs), ber(2, obs), del(2, obs)
100  continue
  print *, ' '
endif
if(choice.ne.4) go to 75
stop
end

```

:::::::::::::

Makefile

:::::::::::::

objects=main.o input.o enumpath.o fathom.o models.o relative.o eval.o

reli.o sta

tes.o maxflow.o

.f.o:

    f77 -c \$\*.f

perf:\$(objects)

    f77 \$(objects) -o perf -limsl -L/usr/local/lib

clean:

    rm \*.o

run:\$(objects)

    f77 \$(objects) -o perf -limsl -L/usr/local/lib

    ./perf



## ***Appendix B. Results For the Sample Network***

### ***B.1 Sample Network Description File***

```
7
10
1 2 0.9      4 1
1 3 0.9      3 1
1 4 0.85     5 1
2 5 0.9      4 1
3 4 0.95     4 2
3 6 0.9      3 1
4 5 0.95     4 2
4 7 0.85     5 1
5 7 0.9      4 1
6 7 0.9      3 1
```

## ***B.2 Sample Network Path Enumeration***

### Depth First Path Enumeration

There are a total of 6 s-t paths.

There are 3 nodes in path number 1  
1 4 7

There are 4 nodes in path number 2  
1 4 5 7

There are 4 nodes in path number 3  
1 3 6 7

There are 4 nodes in path number 4  
1 3 4 7

There are 5 nodes in path number 5  
1 3 4 5 7

There are 4 nodes in path number 6  
1 2 5 7

### B.3 Sample Network Program Output

OPTIONS:

- 1) Obtain dynamic performance bounds
- 2) Investigate steady state performance bounds
- 3) Estimate vector time-series performance data
- 4) Exit this program

Enter 1, 2, 3, or 4 to choose an option.

1

GUIDELINES FOR SETTING PERIOD AND UTILIZATION LEVEL

All of the dynamic models will require a period to be set.  
This period should be greater than the longest path.  
The length of the longest path is 6

The relative performance models call for a utilization level.  
This is a constant value for the level of s-t flow.  
To avoid problems, this value should be kept well under the  
the lower bound on the max expected flow.  
Using an arbitrarily large period, 600, the lower bound on  
expected per unit time flow is 8.69247  
Utilization level should be set with respect to period.  
This value should be period times some level of flow less than  
the stated lower bound on expected max flow.  
The following would be a valid parameter selection:

PERIOD = 12

UTILIZATION = 12 \* 4 = 48

Now set parameters according to the guidelines.

Enter a value for the dynamic period.

(must be greater than 6)

10

Enter a value for the utilization level.

(must be less than 86.9247)

18

Below are the bounds on expected network performance  
when period is set to 10  
and per time period flow is 1.80000

Reliability	Throughput	Bit Error Rate	Average Delay
0.946477 0.950711	73.34 88.65	0.277500 0.277500	2.00000 2.00000

Do you want to compute bounds for other parameters?

(Enter y or n)

y

Enter a value for the dynamic period.

(must be greater than 6)

10

Enter a value for the utilization level.

(must be less than 86.9247)

36

Below are the bounds on expected network performance  
 when period is set to 10  
 and per time period flow is 3.60000

Reliability	Throughput	Bit Error Rate	Average Delay
0.946477 0.950711	73.34 88.65	0.276875 0.277500	2.00000 2.09688

Do you want to compute bounds for other parameters?  
 (Enter y or n)

y

Enter a value for the dynamic period.  
 (must be greater than 6)

10

Enter a value for the utilization level.  
 (must be less than 86.9247)

54

Below are the bounds on expected network performance  
 when period is set to 10  
 and per time period flow is 5.40000

Reliability	Throughput	Bit Error Rate	Average Delay
0.946477 0.950711	73.34 88.65	0.274927 0.275616	2.29167 2.39792

Do you want to compute bounds for other parameters?  
 (Enter y or n)

y

Enter a value for the dynamic period.  
 (must be greater than 6)

10

Enter a value for the utilization level.  
 (must be less than 86.9247)

72

Below are the bounds on expected network performance  
 when period is set to 10  
 and per time period flow is 7.20000

Reliability	Throughput	Bit Error Rate	Average Delay
0.946477 0.950711	73.34 88.65	0.273950 0.274468	2.46875 2.54844

Do you want to compute bounds for other parameters?

(Enter y or n)

n

#### OPTIONS:

- 1) Obtain dynamic performance bounds
- 2) Investigate steady state performance bounds
- 3) Estimate vector time-series performance data
- 4) Exit this program

Enter 1, 2, 3, or 4 to choose an option.

3

How many vector time-series to estimate?  
(100 observations max)

100

#### GUIDELINES FOR SETTING PERIOD AND UTILIZATION LEVEL

All of the dynamic models will require a period to be set.  
This period should be greater than the longest path.  
The length of the longest path is 6

The relative performance models call for a utilization level.  
This is a constant value for the level of s-t flow.  
To avoid problems, this value should be kept well under the  
the lower bound on the max expected flow.

Using an arbitrarily large period, 600, the lower bound on  
expected per unit time flow is 8.69247

Utilization level should be set with respect to period.  
This value should be period times some level of flow less than  
the stated lower bound on expected max flow.  
The following would be a valid parameter selection:

PERIOD = 12  
UTILIZATION = 12 \* 4 = 48

Now set parameters according to the guidelines.  
Enter a value for the dynamic period.  
(must be greater than 6)

10  
Enter a value for the utilization level.  
(must be less than 86.9247)

54

Below is the estimated vector time-series data.

Rel	MaxFlow	BER	DEL
0.923384	67.00	0.274626	2.444444
0.987090	79.00	0.275226	2.35185
0.993632	78.00	0.274626	2.444444
0.978397	75.00	0.274626	2.444444
0.978397	75.00	0.274626	2.444444
0.973498	68.00	0.274626	2.444444
0.965967	64.00	0.274626	2.444444
0.965967	64.00	0.274626	2.444444
0.955712	70.00	0.275226	2.35185
0.935732	65.00	0.274626	2.444444
0.935732	65.00	0.274626	2.444444
0.955712	70.00	0.275226	2.35185
0.966420	76.00	0.275826	2.25926
0.989933	83.00	0.275826	2.25926
0.993422	87.00	0.275826	2.25926
0.993422	87.00	0.275826	2.25926
0.983212	90.00	0.275826	2.25926
0.976135	86.00	0.275826	2.25926
0.933083	79.00	0.275826	2.25926
0.950483	77.00	0.275226	2.35185
0.956175	76.00	0.275226	2.35185
0.936514	76.00	0.275226	2.35185
0.936514	76.00	0.275226	2.35185

0.961202	72.00	0.275226	2.35185
0.945207	72.00	0.275226	2.35185
0.954515	76.00	0.275226	2.35185
0.991621	78.00	0.274626	2.44444
0.991621	78.00	0.274626	2.44444
0.996169	87.00	0.275226	2.35185
0.996169	87.00	0.275226	2.35185
0.986693	84.00	0.275226	2.35185
0.997884	88.00	0.275226	2.35185
0.996624	84.00	0.275226	2.35185
0.990337	79.00	0.274626	2.44444
0.990337	79.00	0.274626	2.44444
0.953171	81.00	0.275226	2.35185
0.948121	82.00	0.275826	2.25926
0.910953	75.00	0.275826	2.25926
0.859324	69.00	0.275226	2.35185
0.829118	64.00	0.274626	2.44444
0.824540	60.00	0.274024	2.53704
0.781017	60.00	0.274024	2.53704
0.848389	65.00	0.274626	2.44444
0.880336	68.00	0.274626	2.44444
0.882138	69.00	0.274626	2.44444
0.954180	71.00	0.274024	2.53704
0.988724	72.00	0.274024	2.53704
0.982360	70.00	0.274625	2.44444
0.990745	78.00	0.275226	2.35185
0.995341	83.00	0.275826	2.25926
0.990745	78.00	0.275226	2.35185
0.933330	69.00	0.274626	2.44444
0.935863	72.00	0.274626	2.44444
0.910865	72.00	0.274626	2.44444
0.925990	76.00	0.275226	2.35185
0.974557	83.00	0.275226	2.35185
0.988264	83.00	0.275226	2.35185
0.947818	79.00	0.275226	2.35185
0.958480	79.00	0.275226	2.35185
0.975336	88.00	0.275826	2.25926
0.984765	93.00	0.276425	2.16667
0.991384	93.00	0.276425	2.16667
0.993461	97.00	0.276425	2.16667
0.978439	90.00	0.276425	2.16667
0.970015	90.00	0.276425	2.16667
0.978908	90.00	0.276425	2.16667
0.983552	86.00	0.276425	2.16667
0.955800	81.00	0.275826	2.25926
0.928563	77.00	0.275826	2.25926
0.928563	77.00	0.275826	2.25926
0.928563	77.00	0.275826	2.25926
0.980071	84.00	0.275826	2.25926
0.977782	79.00	0.275226	2.35185
0.967271	75.00	0.274626	2.44444
0.984471	76.00	0.274625	2.44444
0.935519	69.00	0.274626	2.44444
0.928533	70.00	0.274626	2.44444
0.928533	70.00	0.274626	2.44444
0.928533	70.00	0.274626	2.44444
0.917622	67.00	0.274626	2.44444
0.874136	63.00	0.274626	2.44444
0.891612	71.00	0.275226	2.35185
0.911370	79.00	0.275826	2.25926

0.933966	82.00	0.275826	2.25926
0.919482	85.00	0.275826	2.25926
0.925676	90.00	0.276425	2.16667
0.870191	81.00	0.275826	2.25926
0.924583	80.00	0.275826	2.25926
0.969395	84.00	0.275826	2.25926
0.944536	80.00	0.275826	2.25926
0.973126	80.00	0.275826	2.25926
0.950173	75.00	0.275226	2.35185
0.884232	68.00	0.275226	2.35185
0.853615	63.00	0.274626	2.44444
0.901601	67.00	0.274626	2.44444
0.916299	76.00	0.275226	2.35185
0.918914	69.00	0.275226	2.35185
0.953088	73.00	0.275226	2.35185
0.967700	73.00	0.275226	2.35185
0.950483	72.00	0.274626	2.44444

OPTIONS:

- 1) Obtain dynamic performance bounds
- 2) Investigate steady state performance bounds
- 3) Estimate vector time-series performance data
- 4) Exit this program

Enter 1, 2, 3, or 4 to choose an option.

4

## ***Appendix C. Results For Network A***

### ***C.1 Network Description File For the Original Network A***

```
35
46
1 34 1.0 99999 0
1 2 1.0 99999 0
1 3 1.0 99999 0
1 4 1.0 99999 0
34 12 1.0 1200 1
34 13 1.0 1200 1
34 14 1.0 1200 1
2 20 0.3 99999 0
20 5 0.3 1200 1
20 14 0.6 1200 1
3 21 0.7 99999 0
21 9 1.0 1200 1
21 11 1.0 1200 1
4 22 0.5 99999 0
22 14 1.0 1200 1
5 23 0.8 99999 0
23 10 0.6 1200 1
23 11 0.7 1200 1
6 14 0.6 4800 1
7 24 0.3 99999 0
24 14 0.6 4800 1
8 25 0.7 99999 0
25 14 0.3 4800 1
9 26 0.5 99999 0
26 15 1.0 4800 1
10 27 0.8 99999 0
27 15 0.6 4800 1
11 15 1.0 4800 1
12 28 0.3 99999 0
28 15 0.7 4800 1
13 29 0.7 99999 0
29 15 1.0 4800 1
14 30 0.5 99999 0
30 17 0.3 4800 1
30 18 0.6 4800 1
30 19 0.6 4800 1
15 31 0.8 99999 0
31 6 0.3 4800 1
31 7 0.6 4800 1
31 8 0.7 4800 1
17 32 0.7 99999 0
32 16 0.7 4800 1
18 33 0.3 99999 0
33 16 0.6 4800 1
19 16 0.3 4800 1
16 35 0.8 99999 0
```



## ***C.2 Path Enumeration For the Original Network A***

### Depth First Path Enumeration

There are a total of 63 s-t paths.

There are 8 nodes in path number 1  
1 4 22 14 30 19 16 35

There are 9 nodes in path number 2  
1 4 22 14 30 18 33 16 35

There are 9 nodes in path number 3  
1 4 22 14 30 17 32 16 35

There are 13 nodes in path number 4  
1 3 21 11 15 31 8 25 14 30 19 16 35

There are 14 nodes in path number 5  
1 3 21 11 15 31 8 25 14 30 18 33 16 35

There are 14 nodes in path number 6  
1 3 21 11 15 31 8 25 14 30 17 32 16 35

There are 13 nodes in path number 7  
1 3 21 11 15 31 7 24 14 30 19 16 35

There are 14 nodes in path number 8  
1 3 21 11 15 31 7 24 14 30 18 33 16 35

There are 14 nodes in path number 9  
1 3 21 11 15 31 7 24 14 30 17 32 16 35

There are 12 nodes in path number 10  
1 3 21 11 15 31 6 14 30 19 16 35

There are 13 nodes in path number 11  
1 3 21 11 15 31 6 14 30 18 33 16 35

There are 13 nodes in path number 12  
1 3 21 11 15 31 6 14 30 17 32 16 35

There are 14 nodes in path number 13  
1 3 21 9 26 15 31 8 25 14 30 19 16 35

There are 15 nodes in path number 14  
1 3 21 9 26 15 31 8 25 14 30 18 33 16 35

There are 15 nodes in path number 15  
1 3 21 9 26 15 31 8 25 14 30 17 32 16 35

There are 14 nodes in path number 16  
1 3 21 9 26 15 31 7 24 14 30 19 16 35

There are 15 nodes in path number 17  
1 3 21 9 26 15 31 7 24 14 30 18 33 16 35

There are 15 nodes in path number 18  
1 3 21 9 26 15 31 7 24 14 30 17 32 16 35

There are 13 nodes in path number 19  
1 3 21 9 26 15 31 6 14 30 19 16 35

There are 14 nodes in path number 20  
1 3 21 9 26 15 31 6 14 30 18 33 16 35

There are 14 nodes in path number 21  
1 3 21 9 26 15 31 6 14 30 17 32 16 35

There are 8 nodes in path number 22  
1 2 20 14 30 19 16 35

There are 9 nodes in path number 23  
1 2 20 14 30 18 33 16 35

There are 9 nodes in path number 24  
1 2 20 14 30 17 32 16 35

There are 15 nodes in path number 25  
1 2 20 5 23 11 15 31 8 25 14 30 19 16 35

There are 16 nodes in path number 26  
1 2 20 5 23 11 15 31 8 25 14 30 18 33 16 35

There are 16 nodes in path number 27  
1 2 20 5 23 11 15 31 8 25 14 30 17 32 16 35

There are 15 nodes in path number 28  
1 2 20 5 23 11 15 31 7 24 14 30 19 16 35

There are 16 nodes in path number 29  
1 2 20 5 23 11 15 31 7 24 14 30 18 33 16 35

There are 16 nodes in path number 30  
1 2 20 5 23 11 15 31 7 24 14 30 17 32 16 35

There are 14 nodes in path number 31  
1 2 20 5 23 11 15 31 6 14 30 19 16 35

There are 15 nodes in path number 32  
1 2 20 5 23 11 15 31 6 14 30 18 33 16 35

There are 15 nodes in path number 33  
1 2 20 5 23 11 15 31 6 14 30 17 32 16 35

There are 16 nodes in path number 34  
1 2 20 5 23 10 27 15 31 8 25 14 30 19 16 35

There are 17 nodes in path number 35  
1 2 20 5 23 10 27 15 31 8 25 14 30 18 33 16 35

There are 17 nodes in path number 36  
1 2 20 5 23 10 27 15 31 8 25 14 30 17 32 16 35

There are 16 nodes in path number 37  
1 2 20 5 23 10 27 15 31 7 24 14 30 19 16 35

There are 17 nodes in path number 38  
1 2 20 5 23 10 27 15 31 7 24 14 30 18 33 16 35

There are 17 nodes in path number 39  
1 2 20 5 23 10 27 15 31 7 24 14 30 17 32 16 35

There are 15 nodes in path number 40  
1 2 20 5 23 10 27 15 31 6 14 30 19 16 35

There are 16 nodes in path number 41  
1 2 20 5 23 10 27 15 31 6 14 30 18 33 16 35

There are 16 nodes in path number 42  
1 2 20 5 23 10 27 15 31 6 14 30 17 32 16 35

There are 7 nodes in path number 43  
1 34 14 30 19 16 35

There are 8 nodes in path number 44  
1 34 14 30 18 33 16 35

There are 8 nodes in path number 45  
1 34 14 30 17 32 16 35

There are 13 nodes in path number 46  
1 34 13 29 15 31 8 25 14 30 19 16 35

There are 14 nodes in path number 47  
1 34 13 29 15 31 8 25 14 30 18 33 16 35

There are 14 nodes in path number 48  
1 34 13 29 15 31 8 25 14 30 17 32 16 35

There are 13 nodes in path number 49  
1 34 13 29 15 31 7 24 14 30 19 16 35

There are 14 nodes in path number 50  
1 34 13 29 15 31 7 24 14 30 18 33 16 35

There are 14 nodes in path number 51  
1 34 13 29 15 31 7 24 14 30 17 32 16 35

There are 12 nodes in path number 52  
1 34 13 29 15 31 6 14 30 19 16 35

There are 13 nodes in path number 53  
1 34 13 29 15 31 6 14 30 18 33 16 35

There are 13 nodes in path number 54  
1 34 13 29 15 31 6 14 30 17 32 16 35

There are 13 nodes in path number 55  
1 34 12 28 15 31 8 25 14 30 19 16 35

There are 14 nodes in path number 56  
1 34 12 28 15 31 8 25 14 30 18 33 16 35

There are 14 nodes in path number 57  
1 34 12 28 15 31 8 25 14 30 17 32 16 35

There are 13 nodes in path number 58  
1 34 12 28 15 31 7 24 14 30 19 16 35

There are 14 nodes in path number 59  
1 34 12 28 15 31 7 24 14 30 18 33 16 35

There are 14 nodes in path number 60  
1 34 12 28 15 31 7 24 14 30 17 32 16 35

There are 12 nodes in path number 61  
1 34 12 28 15 31 6 14 30 19 16 35

There are 13 nodes in path number 62  
1 34 12 28 15 31 6 14 30 18 33 16 35

There are 13 nodes in path number 63  
1 34 12 28 15 31 6 14 30 17 32 16 35

### C.3 Program Output For the Original Network A

#### OPTIONS:

- 1) Obtain dynamic performance bounds
- 2) Investigate steady state performance bounds
- 3) Estimate vector time-series performance data
- 4) Exit this program

Enter 1, 2, 3, or 4 to choose an option.

2

For this application utilization level will be assumed to be at the lower bound for max flow.

Reliability		Throughput		Bit Error Rate		Average Delay	
0.000000	0.000000	127.04	3816.00	0.964000	0.973055	3.00000	3.23024
0.000000	0.000000	147.06	4788.00	0.964000	0.976432	3.00000	3.30817
0.000000	0.000000	153.74	5112.00	0.964000	0.977228	3.00000	3.32964
0.000000	0.000000	157.07	5274.00	0.964000	0.977582	3.00000	3.33969
0.000000	0.000000	159.07	5371.20	0.964000	0.977783	3.00000	3.34552
0.000000	0.000000	160.41	5436.00	0.964000	0.977912	3.00000	3.34932
0.000000	0.000000	161.36	5482.29	0.964000	0.978002	3.00000	3.35200
0.000000	0.000000	162.08	5517.00	0.964000	0.978068	3.00000	3.35399
0.000000	0.000000	162.63	5544.00	0.964000	0.978119	3.00000	3.35552
0.000000	0.000000	163.08	5565.60	0.964000	0.978159	3.00000	3.35674
0.000000	0.000000	163.44	5583.27	0.964000	0.978192	3.00000	3.35773
0.000000	0.000000	163.75	5598.00	0.964000	0.978219	3.00000	3.35856
0.000000	0.000000	164.00	5610.46	0.964000	0.978242	3.00000	3.35925
0.000000	0.000000	164.22	5621.14	0.964000	0.978261	3.00000	3.35985
0.000000	0.000000	164.41	5630.40	0.964000	0.978278	3.00000	3.36036
0.000000	0.000000	164.58	5638.50	0.964000	0.978292	3.00000	3.36081
0.000000	0.000000	164.73	5645.65	0.964000	0.978305	3.00000	3.36120
0.000000	0.000000	164.86	5652.00	0.964000	0.978317	3.00000	3.36156
0.000000	0.000000	164.97	5657.68	0.964000	0.978327	3.00000	3.36187
0.000000	0.000000	165.08	5662.80	0.964000	0.978336	3.00000	3.36215
0.000000	0.000000	165.18	5667.43	0.964000	0.978344	3.00000	3.36240
0.000000	0.000000	165.26	5671.64	0.964000	0.978351	3.00000	3.36264
0.000000	0.000000	165.34	5675.48	0.964000	0.978358	3.00000	3.36285
0.000000	0.000000	165.41	5679.00	0.964000	0.978365	3.00000	3.36304
0.000000	0.000000	165.48	5682.24	0.964000	0.978370	3.00000	3.36322
0.000000	0.000000	165.54	5685.23	0.964000	0.978376	3.00000	3.36338
0.000000	0.000000	165.60	5688.00	0.964000	0.978381	3.00000	3.36353
0.000000	0.000000	165.65	5690.57	0.964000	0.978385	3.00000	3.36367
0.000000	0.000000	165.70	5692.97	0.964000	0.978389	3.00000	3.36380
0.000000	0.000000	165.75	5695.20	0.964000	0.978393	3.00000	3.36392
0.000000	0.000000	165.79	5697.29	0.964000	0.978397	3.00000	3.36404
0.000000	0.000000	165.83	5699.25	0.964000	0.978400	3.00000	3.36414
0.000000	0.000000	165.87	5701.09	0.964000	0.978403	3.00000	3.36424
0.000000	0.000000	165.90	5702.82	0.964000	0.978407	3.00000	3.36434
0.000000	0.000000	165.94	5704.46	0.964000	0.978409	3.00000	3.36443
0.000000	0.000000	165.97	5706.00	0.964000	0.978412	3.00000	3.36451
0.000000	0.000000	166.00	5707.46	0.964000	0.978415	3.00000	3.36459
0.000000	0.000000	166.03	5708.84	0.964000	0.978417	3.00000	3.36467
0.000000	0.000000	166.06	5710.15	0.964000	0.978419	3.00000	3.36474
0.000000	0.000000	166.08	5711.40	0.964000	0.978422	3.00000	3.36481
0.000000	0.000000	166.11	5712.59	0.964000	0.978424	3.00000	3.36487
0.000000	0.000000	166.13	5713.71	0.964000	0.978426	3.00000	3.36493
0.000000	0.000000	166.15	5714.79	0.964000	0.978427	3.00000	3.36499

0.000000	0.000000	166.17	5715.82	0.964000	0.978429	3.00000	3.36505
0.000000	0.000000	166.19	5716.80	0.964000	0.978431	3.00000	3.36510
0.000000	0.000000	166.21	5717.74	0.964000	0.978433	3.00000	3.36515
0.000000	0.000000	166.23	5718.64	0.964000	0.978434	3.00000	3.36520
0.000000	0.000000	166.25	5719.50	0.964000	0.978436	3.00000	3.36525
0.000000	0.000000	166.26	5720.33	0.964000	0.978437	3.00000	3.36529
0.000000	0.000000	166.28	5721.12	0.964000	0.978438	3.00000	3.36533
0.000000	0.000000	166.30	5721.88	0.964000	0.978440	3.00000	3.36537
0.000000	0.000000	166.31	5722.62	0.964000	0.978441	3.00000	3.36541
0.000000	0.000000	166.33	5723.32	0.964000	0.978442	3.00000	3.36545
0.000000	0.000000	166.34	5724.00	0.964000	0.978443	3.00000	3.36549
0.000000	0.000000	166.35	5724.65	0.964000	0.978445	3.00000	3.36552
0.000000	0.000000	166.37	5725.29	0.964000	0.978446	3.00000	3.36556
0.000000	0.000000	166.38	5725.89	0.964000	0.978447	3.00000	3.36559
0.000000	0.000000	166.39	5726.48	0.964000	0.978448	3.00000	3.36562
0.000000	0.000000	166.40	5727.05	0.964000	0.978449	3.00000	3.36565
0.000000	0.000000	166.41	5727.60	0.964000	0.978450	3.00000	3.36568
0.000000	0.000000	166.43	5728.13	0.964000	0.978451	3.00000	3.36571
0.000000	0.000000	166.44	5728.65	0.964000	0.978452	3.00000	3.36574
0.000000	0.000000	166.45	5729.14	0.964000	0.978452	3.00000	3.36577
0.000000	0.000000	166.46	5729.62	0.964000	0.978453	3.00000	3.36579
0.000000	0.000000	166.47	5730.09	0.964000	0.978454	3.00000	3.36582
0.000000	0.000000	166.48	5730.55	0.964000	0.978455	3.00000	3.36584
0.000000	0.000000	166.48	5730.98	0.964000	0.978456	3.00000	3.36587
0.000000	0.000000	166.49	5731.41	0.964000	0.978456	3.00000	3.36589
0.000000	0.000000	166.50	5731.83	0.964000	0.978457	3.00000	3.36591
0.000000	0.000000	166.51	5732.23	0.964000	0.978458	3.00000	3.36593
0.000000	0.000000	166.52	5732.62	0.964000	0.978458	3.00000	3.36595
0.000000	0.000000	166.53	5733.00	0.964000	0.978459	3.00000	3.36598
0.000000	0.000000	166.53	5733.37	0.964000	0.978460	3.00000	3.36600
0.000000	0.000000	166.54	5733.73	0.964000	0.978460	3.00000	3.36601
0.000000	0.000000	166.55	5734.08	0.964000	0.978461	3.00000	3.36603
0.000000	0.000000	166.55	5734.42	0.964000	0.978462	3.00000	3.36605
0.000000	0.000000	166.56	5734.75	0.964000	0.978462	3.00000	3.36607
0.000000	0.000000	166.57	5735.08	0.964000	0.978463	3.00000	3.36609
0.000000	0.000000	166.57	5735.39	0.964000	0.978463	3.00000	3.36610
0.000000	0.000000	166.58	5735.70	0.964000	0.978464	3.00000	3.36612
0.000000	0.000000	166.59	5736.00	0.964000	0.978464	3.00000	3.36614
0.000000	0.000000	166.59	5736.29	0.964000	0.978465	3.00000	3.36615
0.000000	0.000000	166.60	5736.58	0.964000	0.978465	3.00000	3.36617
0.000000	0.000000	166.61	5736.86	0.964000	0.978466	3.00000	3.36618
0.000000	0.000000	166.61	5737.13	0.964000	0.978466	3.00000	3.36620
0.000000	0.000000	166.62	5737.40	0.964000	0.978467	3.00000	3.36621
0.000000	0.000000	166.62	5737.66	0.964000	0.978467	3.00000	3.36623
0.000000	0.000000	166.63	5737.91	0.964000	0.978468	3.00000	3.36624
0.000000	0.000000	166.63	5738.16	0.964000	0.978468	3.00000	3.36625
0.000000	0.000000	166.64	5738.40	0.964000	0.978469	3.00000	3.36627
0.000000	0.000000	166.64	5738.64	0.964000	0.978469	3.00000	3.36628
0.000000	0.000000	166.65	5738.87	0.964000	0.978469	3.00000	3.36629
0.000000	0.000000	166.65	5739.10	0.964000	0.978470	3.00000	3.36630
0.000000	0.000000	166.66	5739.32	0.964000	0.978470	3.00000	3.36632
0.000000	0.000000	166.66	5739.54	0.964000	0.978470	3.00000	3.36633
0.000000	0.000000	166.66	5739.75	0.964000	0.978471	3.00000	3.36634
0.000000	0.000000	166.67	5739.96	0.964000	0.978471	3.00000	3.36635
0.000000	0.000000	166.67	5740.16	0.964000	0.978471	3.00000	3.36636
0.000000	0.000000	166.68	5740.36	0.964000	0.978472	3.00000	3.36637
0.000000	0.000000	166.68	5740.56	0.964000	0.978472	3.00000	3.36638

The steady state per time unit performance estimates  
at an arbitrarily large period 1000 are below.

Reliability		Throughput		Bit Error Rate		Average Delay	
0.000000	0.000000	166.68	5740.56	0.964000	0.978472	3.00000	3.36638

OPTIONS:

- 1) Obtain dynamic performance bounds
- 2) Investigate steady state performance bounds
- 3) Estimate vector time-series performance data
- 4) Exit this program

Enter 1, 2, 3, or 4 to choose an option.

4

#### ***C.4 Network Description File For the New Network A***

```
20
31
1 19 1.000 99999 0
1 2 1.000 99999 0
1 3 1.000 99999 0
1 4 1.000 99999 0
19 12 0.914 1200 1
19 13 0.987 1200 1
19 14 0.945 1200 1
2 5 0.927 1200 1
2 14 0.911 1200 1
3 9 0.980 1200 1
3 11 0.996 1200 1
4 14 0.963 1200 1
5 10 0.944 1200 1
5 11 0.936 1200 1
6 14 0.909 4800 1
7 14 0.940 4800 1
8 14 0.971 4800 1
9 15 0.927 4800 1
10 15 0.955 4800 1
11 15 0.963 4800 1
12 15 0.919 4800 1
13 15 0.977 4800 1
14 17 0.945 4800 1
14 18 0.954 4800 1
14 16 0.932 4800 1
15 6 0.990 4800 1
15 7 0.951 4800 1
15 8 0.946 4800 1
17 20 0.907 4800 1
18 20 0.934 4800 1
16 20 0.918 4800 1
```



## ***C.5 Path Enumeration For the New Network A***

### Depth First Path Enumeration

There are a total of 63 s-t paths.

There are 5 nodes in path number 1  
1 4 14 16 20

There are 5 nodes in path number 2  
1 4 14 18 20

There are 5 nodes in path number 3  
1 4 14 17 20

There are 8 nodes in path number 4  
1 3 11 15 8 14 16 20

There are 8 nodes in path number 5  
1 3 11 15 8 14 18 20

There are 8 nodes in path number 6  
1 3 11 15 8 14 17 20

There are 8 nodes in path number 7  
1 3 11 15 7 14 16 20

There are 8 nodes in path number 8  
1 3 11 15 7 14 18 20

There are 8 nodes in path number 9  
1 3 11 15 7 14 17 20

There are 8 nodes in path number 10  
1 3 11 15 6 14 16 20

There are 8 nodes in path number 11  
1 3 11 15 6 14 18 20

There are 8 nodes in path number 12  
1 3 11 15 6 14 17 20

There are 8 nodes in path number 13  
1 3 9 15 8 14 16 20

There are 8 nodes in path number 14  
1 3 9 15 8 14 18 20

There are 8 nodes in path number 15  
1 3 9 15 8 14 17 20

There are 8 nodes in path number 16  
1 3 9 15 7 14 16 20

There are 8 nodes in path number 17  
1 3 9 15 7 14 18 20

There are 8 nodes in path number 18  
1 3 9 15 7 14 17 20

There are 8 nodes in path number 19  
1 3 9 15 6 14 16 20

There are 8 nodes in path number 20  
1 3 9 15 6 14 18 20

There are 8 nodes in path number 21  
1 3 9 15 6 14 17 20

There are 5 nodes in path number 22  
1 2 14 16 20

There are 5 nodes in path number 23  
1 2 14 18 20

There are 5 nodes in path number 24  
1 2 14 17 20

There are 9 nodes in path number 25  
1 2 5 11 15 8 14 16 20

There are 9 nodes in path number 26  
1 2 5 11 15 8 14 18 20

There are 9 nodes in path number 27  
1 2 5 11 15 8 14 17 20

There are 9 nodes in path number 28  
1 2 5 11 15 7 14 16 20

There are 9 nodes in path number 29  
1 2 5 11 15 7 14 18 20

There are 9 nodes in path number 30  
1 2 5 11 15 7 14 17 20

There are 9 nodes in path number 31  
1 2 5 11 15 6 14 16 20

There are 9 nodes in path number 32  
1 2 5 11 15 6 14 18 20

There are 9 nodes in path number 33  
1 2 5 11 15 6 14 17 20

There are 9 nodes in path number 34  
1 2 5 10 15 8 14 16 20

There are 9 nodes in path number 35  
1 2 5 10 15 8 14 18 20

There are 9 nodes in path number 36  
1 2 5 10 15 8 14 17 20

There are 9 nodes in path number 37  
1 2 5 10 15 7 14 16 20

There are 9 nodes in path number 38  
1 2 5 10 15 7 14 18 20

There are 9 nodes in path number 39  
 1 2 5 10 15 7 14 17 20

There are 9 nodes in path number 40  
 1 2 5 10 15 6 14 16 20

There are 9 nodes in path number 41  
 1 2 5 10 15 6 14 18 20

There are 9 nodes in path number 42  
 1 2 5 10 15 6 14 17 20

There are 5 nodes in path number 43  
 1 19 14 16 20

There are 5 nodes in path number 44  
 1 19 14 18 20

There are 5 nodes in path number 45  
 1 19 14 17 20

There are 8 nodes in path number 46  
 1 19 13 15 8 14 16 20

There are 8 nodes in path number 47  
 1 19 13 15 8 14 18 20

There are 8 nodes in path number 48  
 1 19 13 15 8 14 17 20

There are 8 nodes in path number 49  
 1 19 13 15 7 14 16 20

There are 8 nodes in path number 50  
 1 19 13 15 7 14 18 20

There are 8 nodes in path number 51  
 1 19 13 15 7 14 17 20

There are 8 nodes in path number 52  
 1 19 13 15 6 14 16 20

There are 8 nodes in path number 53  
 1 19 13 15 6 14 18 20

There are 8 nodes in path number 54  
 1 19 13 15 6 14 17 20

There are 8 nodes in path number 55  
 1 19 12 15 8 14 16 20

There are 8 nodes in path number 56  
 1 19 12 15 8 14 18 20

There are 8 nodes in path number 57  
 1 19 12 15 8 14 17 20

There are 8 nodes in path number 58  
 1 19 12 15 7 14 16 20

There are 8 nodes in path number 59  
1 19 12 15 7 14 18 20

There are 8 nodes in path number 60  
1 19 12 15 7 14 17 20

There are 8 nodes in path number 61  
1 19 12 15 6 14 16 20

There are 8 nodes in path number 62  
1 19 12 15 6 14 18 20

There are 8 nodes in path number 63  
1 19 12 15 6 14 17 20

## C.6 Program Output For the New Network A

OPTIONS:

- 1) Obtain dynamic performance bounds
- 2) Investigate steady state performance bounds
- 3) Estimate vector time-series performance data
- 4) Exit this program

Enter 1, 2, 3, or 4 to choose an option.

OPTIONS:

- 1) Obtain dynamic performance bounds
- 2) Investigate steady state performance bounds
- 3) Estimate vector time-series performance data
- 4) Exit this program

Enter 1, 2, 3, or 4 to choose an option.

1

### GUIDELINES FOR SETTING PERIOD AND UTILIZATION LEVEL

All of the dynamic models will require a period to be set.  
This period should be greater than the longest path.  
The length of the longest path is 7

The relative performance models call for a utilization level.  
This is a constant value for the level of s-t flow.  
To avoid problems, this value should be kept well under the  
the lower bound on the max expected flow.

Using an arbitrarily large period, 700, the lower bound on  
expected per unit time flow is 7252.61

Utilization level should be set with respect to period.  
This value should be period times some level of flow less than  
the stated lower bound on expected max flow.

The following would be a valid parameter selection:

PERIOD = 14

UTILIZATION = 14 \* 3626 = 50764

Now set parameters according to the guidelines.

Enter a value for the dynamic period.  
(must be greater than 7)

10

Enter a value for the utilization level.  
(must be less than 72526.1)

11000

Below are the bounds on expected network performance  
when period is set to 10  
and per time period flow is 1100.00

Reliability	Throughput	Bit Error Rate	Average Delay
0.985409 0.998474	44733.51 54774.00	0.146017 0.183517	3.00000 3.00000

Do you want to compute bounds for other parameters?  
(Enter y or n)

y

Enter a value for the dynamic period.

(must be greater than 7)

10

Enter a value for the utilization level.

(must be less than 72526.1)

22000

Below are the bounds on expected network performance

when period is set to 10

and per time period flow is 2200.00

Reliability	Throughput	Bit Error Rate	Average Delay
0.985409	0.998474	44733.51	54774.00
0.160184	0.197087	3.00000	3.00000

Do you want to compute bounds for other parameters?

(Enter y or n)

y

Enter a value for the dynamic period.

(must be greater than 7)

10

Enter a value for the utilization level.

(must be less than 72526.1)

33000

Below are the bounds on expected network performance

when period is set to 10

and per time period flow is 3300.00

Reliability	Throughput	Bit Error Rate	Average Delay
0.985409	0.998474	44733.51	54774.00
0.181076	0.204342	3.53978	3.80786

Do you want to compute bounds for other parameters?

(Enter y or n)

y

Enter a value for the dynamic period.

(must be greater than 7)

10

Enter a value for the utilization level.

(must be less than 72526.1)

44000

Below are the bounds on expected network performance

when period is set to 10

and per time period flow is 4400.00

Reliability	Throughput	Bit Error Rate	Average Delay
0.985409	0.998474	44733.51	54774.00
0.210971	0.220716	4.15484	4.40954

Do you want to compute bounds for other parameters?

(Enter y or n)

n

OPTIONS:

- 1) Obtain dynamic performance bounds
- 2) Investigate steady state performance bounds
- 3) Estimate vector time-series performance data
- 4) Exit this program

Enter 1, 2, 3, or 4 to choose an option.

4

## ***Appendix D. Results For Network B***

### ***D.1 Network Description File For Network B***

```
28
53
1 27 1.000 99999 0
1 2 1.000 99999 0
1 3 1.000 99999 0
1 4 1.000 99999 0
1 5 1.000 99999 0
1 6 1.000 99999 0
27 7 0.916 150 1
27 8 0.930 200 1
2 8 0.975 750 1
2 9 0.952 750 1
3 7 0.991 200 1
3 9 0.917 750 1
3 16 0.984 150 1
4 16 0.904 200 1
4 24 0.950 600 1
5 14 0.925 1200 1
6 16 0.930 1200 1
6 25 0.971 75 1
6 26 0.946 75 1
7 10 0.902 1200 1
8 10 0.988 1200 1
9 10 0.917 2400 1
10 11 0.911 1200 1
10 12 0.985 1200 1
10 13 0.944 1200 1
11 17 0.969 1200 1
11 18 0.996 75 1
11 19 0.978 1200 1
12 16 0.912 1200 1
12 20 0.903 1200 1
13 16 0.927 600 1
14 15 0.930 1200 1
15 16 0.996 1200 1
16 17 0.954 75 1
16 18 0.965 75 1
16 19 0.929 75 1
16 20 0.983 75 1
16 21 0.977 75 1
16 22 0.945 75 1
16 23 0.918 75 1
16 24 0.990 75 1
16 25 0.971 75 1
16 26 0.954 75 1
17 28 1.000 99999 0
18 28 1.000 99999 0
19 28 1.000 99999 0
20 28 1.000 99999 0
21 28 1.000 99999 0
22 28 1.000 99999 0
23 28 1.000 99999 0
```



24	28	1.000	99999	0
25	28	1.000	99999	0
26	28	1.000	99999	0

## ***D.2 Path Enumeration For Network B***

### Depth First Path Enumeration

There are a total of 187 s-t paths.

There are	4 nodes in path number	1
	1 6 26 28	
There are	4 nodes in path number	2
	1 6 25 28	
There are	5 nodes in path number	3
	1 6 16 26 28	
There are	5 nodes in path number	4
	1 6 16 25 28	
There are	5 nodes in path number	5
	1 6 16 24 28	
There are	5 nodes in path number	6
	1 6 16 23 28	
There are	5 nodes in path number	7
	1 6 16 22 28	
There are	5 nodes in path number	8
	1 6 16 21 28	
There are	5 nodes in path number	9
	1 6 16 20 28	
There are	5 nodes in path number	10
	1 6 16 19 28	
There are	5 nodes in path number	11
	1 6 16 18 28	
There are	5 nodes in path number	12
	1 6 16 17 28	
There are	7 nodes in path number	13
	1 5 14 15 16 26 28	
There are	7 nodes in path number	14
	1 5 14 15 16 25 28	
There are	7 nodes in path number	15
	1 5 14 15 16 24 28	
There are	7 nodes in path number	16
	1 5 14 15 16 23 28	
There are	7 nodes in path number	17
	1 5 14 15 16 22 28	
There are	7 nodes in path number	18
	1 5 14 15 16 21 28	

There are 7 nodes in path number 19  
1 5 14 15 16 20 28

There are 7 nodes in path number 20  
1 5 14 15 16 19 28

There are 7 nodes in path number 21  
1 5 14 15 16 18 28

There are 7 nodes in path number 22  
1 5 14 15 16 17 28

There are 4 nodes in path number 23  
1 4 24 28

There are 5 nodes in path number 24  
1 4 16 26 28

There are 5 nodes in path number 25  
1 4 16 25 28

There are 5 nodes in path number 26  
1 4 16 24 28

There are 5 nodes in path number 27  
1 4 16 23 28

There are 5 nodes in path number 28  
1 4 16 22 28

There are 5 nodes in path number 29  
1 4 16 21 28

There are 5 nodes in path number 30  
1 4 16 20 28

There are 5 nodes in path number 31  
1 4 16 19 28

There are 5 nodes in path number 32  
1 4 16 18 28

There are 5 nodes in path number 33  
1 4 16 17 28

There are 5 nodes in path number 34  
1 3 16 26 28

There are 5 nodes in path number 35  
1 3 16 25 28

There are 5 nodes in path number 36  
1 3 16 24 28

There are 5 nodes in path number 37  
1 3 16 23 28

There are 5 nodes in path number 38  
1 3 16 22 28

There are 5 nodes in path number 39  
 1 3 16 21 28

There are 5 nodes in path number 40  
 1 3 16 20 28

There are 5 nodes in path number 41  
 1 3 16 19 28

There are 5 nodes in path number 42  
 1 3 16 18 28

There are 5 nodes in path number 43  
 1 3 16 17 28

There are 8 nodes in path number 44  
 1 3 9 10 13 16 26 28

There are 8 nodes in path number 45  
 1 3 9 10 13 16 25 28

There are 8 nodes in path number 46  
 1 3 9 10 13 16 24 28

There are 8 nodes in path number 47  
 1 3 9 10 13 16 23 28

There are 8 nodes in path number 48  
 1 3 9 10 13 16 22 28

There are 8 nodes in path number 49  
 1 3 9 10 13 16 21 28

There are 8 nodes in path number 50  
 1 3 9 10 13 16 20 28

There are 8 nodes in path number 51  
 1 3 9 10 13 16 19 28

There are 8 nodes in path number 52  
 1 3 9 10 13 16 18 28

There are 8 nodes in path number 53  
 1 3 9 10 13 16 17 28

There are 7 nodes in path number 54  
 1 3 9 10 12 20 28

There are 8 nodes in path number 55  
 1 3 9 10 12 16 26 28

There are 8 nodes in path number 56  
 1 3 9 10 12 16 25 28

There are 8 nodes in path number 57  
 1 3 9 10 12 16 24 28

There are 8 nodes in path number 58  
 1 3 9 10 12 16 23 28

There are 8 nodes in path number 59  
 1 3 9 10 12 16 22 28  
 There are 8 nodes in path number 60  
 1 3 9 10 12 16 21 28  
 There are 8 nodes in path number 61  
 1 3 9 10 12 16 20 28  
 There are 8 nodes in path number 62  
 1 3 9 10 12 16 19 28  
 There are 8 nodes in path number 63  
 1 3 9 10 12 16 18 28  
 There are 8 nodes in path number 64  
 1 3 9 10 12 16 17 28  
 There are 7 nodes in path number 65  
 1 3 9 10 11 19 28  
 There are 7 nodes in path number 66  
 1 3 9 10 11 18 28  
 There are 7 nodes in path number 67  
 1 3 9 10 11 17 28  
 There are 8 nodes in path number 68  
 1 3 7 10 13 16 26 28  
 There are 8 nodes in path number 69  
 1 3 7 10 13 16 25 28  
 There are 8 nodes in path number 70  
 1 3 7 10 13 16 24 28  
 There are 8 nodes in path number 71  
 1 3 7 10 13 16 23 28  
 There are 8 nodes in path number 72  
 1 3 7 10 13 16 22 28  
 There are 8 nodes in path number 73  
 1 3 7 10 13 16 21 28  
 There are 8 nodes in path number 74  
 1 3 7 10 13 16 20 28  
 There are 8 nodes in path number 75  
 1 3 7 10 13 16 19 28  
 There are 8 nodes in path number 76  
 1 3 7 10 13 16 18 28  
 There are 8 nodes in path number 77  
 1 3 7 10 13 16 17 28  
 There are 7 nodes in path number 78  
 1 3 7 10 12 20 28

There are 8 nodes in path number 79  
     1 3 7 10 12 16 26 28  
  
 There are 8 nodes in path number 80  
     1 3 7 10 12 16 25 28  
  
 There are 8 nodes in path number 81  
     1 3 7 10 12 16 24 28  
  
 There are 8 nodes in path number 82  
     1 3 7 10 12 16 23 28  
  
 There are 8 nodes in path number 83  
     1 3 7 10 12 16 22 28  
  
 There are 8 nodes in path number 84  
     1 3 7 10 12 16 21 28  
  
 There are 8 nodes in path number 85  
     1 3 7 10 12 16 20 28  
  
 There are 8 nodes in path number 86  
     1 3 7 10 12 16 19 28  
  
 There are 8 nodes in path number 87  
     1 3 7 10 12 16 18 28  
  
 There are 8 nodes in path number 88  
     1 3 7 10 12 16 17 28  
  
 There are 7 nodes in path number 89  
     1 3 7 10 11 19 28  
  
 There are 7 nodes in path number 90  
     1 3 7 10 11 18 28  
  
 There are 7 nodes in path number 91  
     1 3 7 10 11 17 28  
  
 There are 8 nodes in path number 92  
     1 2 9 10 13 16 26 28  
  
 There are 8 nodes in path number 93  
     1 2 9 10 13 16 25 28  
  
 There are 8 nodes in path number 94  
     1 2 9 10 13 16 24 28  
  
 There are 8 nodes in path number 95  
     1 2 9 10 13 16 23 28  
  
 There are 8 nodes in path number 96  
     1 2 9 10 13 16 22 28  
  
 There are 8 nodes in path number 97  
     1 2 9 10 13 16 21 28  
  
 There are 8 nodes in path number 98  
     1 2 9 10 13 16 20 28

There are 8 nodes in path number 99  
1 2 9 10 13 16 19 28

There are 8 nodes in path number 100  
1 2 9 10 13 16 18 28

There are 8 nodes in path number 101  
1 2 9 10 13 16 17 28

There are 7 nodes in path number 102  
1 2 9 10 12 20 28

There are 8 nodes in path number 103  
1 2 9 10 12 16 26 28

There are 8 nodes in path number 104  
1 2 9 10 12 16 25 28

There are 8 nodes in path number 105  
1 2 9 10 12 16 24 28

There are 8 nodes in path number 106  
1 2 9 10 12 16 23 28

There are 8 nodes in path number 107  
1 2 9 10 12 16 22 28

There are 8 nodes in path number 108  
1 2 9 10 12 16 21 28

There are 8 nodes in path number 109  
1 2 9 10 12 16 20 28

There are 8 nodes in path number 110  
1 2 9 10 12 16 19 28

There are 8 nodes in path number 111  
1 2 9 10 12 16 18 28

There are 8 nodes in path number 112  
1 2 9 10 12 16 17 28

There are 7 nodes in path number 113  
1 2 9 10 11 19 28

There are 7 nodes in path number 114  
1 2 9 10 11 18 28

There are 7 nodes in path number 115  
1 2 9 10 11 17 28

There are 8 nodes in path number 116  
1 2 8 10 13 16 26 28

There are 8 nodes in path number 117  
1 2 8 10 13 16 25 28

There are 8 nodes in path number 118  
1 2 8 10 13 16 24 28

There are 8 nodes in path number 119  
1 2 8 10 13 16 23 28

There are 8 nodes in path number 120  
1 2 8 10 13 16 22 28

There are 8 nodes in path number 121  
1 2 8 10 13 16 21 28

There are 8 nodes in path number 122  
1 2 8 10 13 16 20 28

There are 8 nodes in path number 123  
1 2 8 10 13 16 19 28

There are 8 nodes in path number 124  
1 2 8 10 13 16 18 28

There are 8 nodes in path number 125  
1 2 8 10 13 16 17 28

There are 7 nodes in path number 126  
1 2 8 10 12 20 28

There are 8 nodes in path number 127  
1 2 8 10 12 16 26 28

There are 8 nodes in path number 128  
1 2 8 10 12 16 25 28

There are 8 nodes in path number 129  
1 2 8 10 12 16 24 28

There are 8 nodes in path number 130  
1 2 8 10 12 16 23 28

There are 8 nodes in path number 131  
1 2 8 10 12 16 22 28

There are 8 nodes in path number 132  
1 2 8 10 12 16 21 28

There are 8 nodes in path number 133  
1 2 8 10 12 16 20 28

There are 8 nodes in path number 134  
1 2 8 10 12 16 19 28

There are 8 nodes in path number 135  
1 2 8 10 12 16 18 28

There are 8 nodes in path number 136  
1 2 8 10 12 16 17 28

There are 7 nodes in path number 137  
1 2 8 10 11 19 28

There are 7 nodes in path number 138  
1 2 8 10 11 18 28



There are 7 nodes in path number 139  
1 2 8 10 11 17 28

There are 8 nodes in path number 140  
1 27 8 10 13 16 26 28

There are 8 nodes in path number 141  
1 27 8 10 13 16 25 28

There are 8 nodes in path number 142  
1 27 8 10 13 16 24 28

There are 8 nodes in path number 143  
1 27 8 10 13 16 23 28

There are 8 nodes in path number 144  
1 27 8 10 13 16 22 28

There are 8 nodes in path number 145  
1 27 8 10 13 16 21 28

There are 8 nodes in path number 146  
1 27 8 10 13 16 20 28

There are 8 nodes in path number 147  
1 27 8 10 13 16 19 28

There are 8 nodes in path number 148  
1 27 8 10 13 16 18 28

There are 8 nodes in path number 149  
1 27 8 10 13 16 17 28

There are 7 nodes in path number 150  
1 27 8 10 12 20 28

There are 8 nodes in path number 151  
1 27 8 10 12 16 26 28

There are 8 nodes in path number 152  
1 27 8 10 12 16 25 28

There are 8 nodes in path number 153  
1 27 8 10 12 16 24 28

There are 8 nodes in path number 154  
1 27 8 10 12 16 23 28

There are 8 nodes in path number 155  
1 27 8 10 12 16 22 28

There are 8 nodes in path number 156  
1 27 8 10 12 16 21 28

There are 8 nodes in path number 157  
1 27 8 10 12 16 20 28

There are 8 nodes in path number 158  
1 27 8 10 12 16 19 28

There are 8 nodes in path number 159  
1 27 8 10 12 16 18 28

There are 8 nodes in path number 160  
1 27 8 10 12 16 17 28

There are 7 nodes in path number 161  
1 27 8 10 11 19 28

There are 7 nodes in path number 162  
1 27 8 10 11 18 28

There are 7 nodes in path number 163  
1 27 8 10 11 17 28

There are 8 nodes in path number 164  
1 27 7 10 13 16 26 28

There are 8 nodes in path number 165  
1 27 7 10 13 16 25 28

There are 8 nodes in path number 166  
1 27 7 10 13 16 24 28

There are 8 nodes in path number 167  
1 27 7 10 13 16 23 28

There are 8 nodes in path number 168  
1 27 7 10 13 16 22 28

There are 8 nodes in path number 169  
1 27 7 10 13 16 21 28

There are 8 nodes in path number 170  
1 27 7 10 13 16 20 28

There are 8 nodes in path number 171  
1 27 7 10 13 16 19 28

There are 8 nodes in path number 172  
1 27 7 10 13 16 18 28

There are 8 nodes in path number 173  
1 27 7 10 13 16 17 28

There are 7 nodes in path number 174  
1 27 7 10 12 20 28

There are 8 nodes in path number 175  
1 27 7 10 12 16 26 28

There are 8 nodes in path number 176  
1 27 7 10 12 16 25 28

There are 8 nodes in path number 177  
1 27 7 10 12 16 24 28

There are 8 nodes in path number 178  
1 27 7 10 12 16 23 28

There are 8 nodes in path number 179  
1 27 7 10 12 16 22 28

There are 8 nodes in path number 180  
1 27 7 10 12 16 21 28

There are 8 nodes in path number 181  
1 27 7 10 12 16 20 28

There are 8 nodes in path number 182  
1 27 7 10 12 16 19 28

There are 8 nodes in path number 183  
1 27 7 10 12 16 18 28

There are 8 nodes in path number 184  
1 27 7 10 12 16 17 28

There are 7 nodes in path number 185  
1 27 7 10 11 19 28

There are 7 nodes in path number 186  
1 27 7 10 11 18 28

There are 7 nodes in path number 187  
1 27 7 10 11 17 28

### D.3 Program Output For Network B

OPTIONS:

- 1) Obtain dynamic performance bounds
- 2) Investigate steady state performance bounds
- 3) Estimate vector time-series performance data
- 4) Exit this program

Enter 1, 2, 3, or 4 to choose an option.

OPTIONS:

- 1) Obtain dynamic performance bounds
- 2) Investigate steady state performance bounds
- 3) Estimate vector time-series performance data
- 4) Exit this program

Enter 1, 2, 3, or 4 to choose an option.

1

#### GUIDELINES FOR SETTING PERIOD AND UTILIZATION LEVEL

All of the dynamic models will require a period to be set.  
This period should be greater than the longest path.  
The length of the longest path is 5

The relative performance models call for a utilization level.  
This is a constant value for the level of s-t flow.  
To avoid problems, this value should be kept well under the  
the lower bound on the max expected flow.

Using an arbitrarily large period, 500, the lower bound on  
expected per unit time flow is 3301.77

Utilization level should be set with respect to period.  
This value should be period times some level of flow less than  
the stated lower bound on expected max flow.

The following would be a valid parameter selection:

PERIOD = 10

UTILIZATION = 10 \* 1650 = 16500

Now set parameters according to the guidelines.

Enter a value for the dynamic period.

(must be greater than 5)

10

Enter a value for the utilization level.

(must be less than 33017.7)

7500

Below are the bounds on expected network performance

when period is set to 10  
and per time period flow is 750.000

Reliability	Throughput	Bit Error Rate	Average Delay
0.964520	1.000000	26697.26 28845.90	0.047239 0.051629

Do you want to compute bounds for other parameters?

(Enter y or n)

y

Enter a value for the dynamic period.  
(must be greater than 5)

10

Enter a value for the utilization level.  
(must be less than 33017.7)

15000

Below are the bounds on expected network performance  
when period is set to 10  
and per time period flow is 1500.00

Reliability	Throughput	Bit Error Rate	Average Delay
0.964520	1.000000	26697.26 28845.90	0.080083 0.097407
			1.70971 1.76051

Do you want to compute bounds for other parameters?  
(Enter y or n)

y

Enter a value for the dynamic period.  
(must be greater than 5)

10

Enter a value for the utilization level.  
(must be less than 33017.7)

22500

Below are the bounds on expected network performance  
when period is set to 10  
and per time period flow is 2250.00

Reliability	Throughput	Bit Error Rate	Average Delay
0.964520	1.000000	26697.26 28845.90	0.118194 0.147769
			2.47314 2.50701

Do you want to compute bounds for other parameters?  
(Enter y or n)

y

Enter a value for the dynamic period.  
(must be greater than 5)

10

Enter a value for the utilization level.  
(must be less than 33017.7)

30000

Network failure  
Network failure

Below are the bounds on expected network performance  
when period is set to 10  
and per time period flow is 3000.00

Reliability	Throughput	Bit Error Rate	Average Delay
0.964520	1.000000	26697.26 28845.90	0.140185 0.193784
			2.74173 2.80429

Do you want to compute bounds for other parameters?

(Enter y or n)

n

OPTIONS:

- 1) Obtain dynamic performance bounds
- 2) Investigate steady state performance bounds
- 3) Estimate vector time-series performance data
- 4) Exit this program

Enter 1, 2, 3, or 4 to choose an option.

4

## *Appendix E. Results For Network C*

### *E.1 Network Description File For Network C*

```
41
77
1 40 1.000 99999 0
1 2 1.000 99999 0
1 3 1.000 99999 0
1 4 1.000 99999 0
1 5 1.000 99999 0
1 6 1.000 99999 0
40 11 0.974 1200 1
2 11 0.913 1200 1
3 7 0.948 300 1
4 11 0.979 1200 1
5 8 0.921 1200 1
6 9 0.900 1200 1
7 10 0.937 300 1
8 11 0.983 1200 1
9 11 0.916 1200 1
10 11 0.944 300 1
11 12 0.900 9600 1
11 23 0.968 75 1
11 24 0.984 75 1
11 38 0.990 1200 1
11 39 0.914 1200 1
12 13 0.940 4800 1
12 14 0.934 4800 1
12 15 0.970 4800 1
12 16 0.988 4800 1
12 17 0.915 4800 1
12 18 0.959 2400 1
12 19 0.907 4800 1
12 20 0.911 4800 1
12 21 0.996 4800 1
12 22 0.935 2400 1
13 23 0.963 4800 1
13 24 0.907 4800 1
13 25 0.976 2400 1
13 26 0.949 1200 1
13 27 0.971 1200 1
13 28 0.980 1200 1
13 29 0.966 1200 1
14 23 0.949 4800 1
14 24 0.992 4800 1
14 25 0.924 2400 1
14 26 0.905 1200 1
14 27 0.958 1200 1
14 28 0.977 1200 1
14 29 0.900 1200 1
15 31 0.915 2400 1
15 32 0.931 1200 1
16 30 0.963 300 1
16 31 0.926 2400 1
16 32 0.949 1200 1
```

16	33	0.985	300	1
16	36	0.979	2400	1
17	30	0.906	1200	1
17	33	0.998	300	1
17	34	0.930	300	1
18	38	0.943	1200	1
19	39	0.977	1200	1
20	35	0.904	2400	1
21	36	0.951	2400	1
22	37	0.914	600	1
23	41	1.000	99999	0
24	41	1.000	99999	0
25	41	1.000	99999	0
26	41	1.000	99999	0
27	41	1.000	99999	0
28	41	1.000	99999	0
29	41	1.000	99999	0
30	41	1.000	99999	0
31	41	1.000	99999	0
32	41	1.000	99999	0
33	41	1.000	99999	0
34	41	1.000	99999	0
35	41	1.000	99999	0
36	41	1.000	99999	0
37	41	1.000	99999	0
38	41	1.000	99999	0
39	41	1.000	99999	0



## ***E.2 Path Enumeration For Network C***

### Depth First Path Enumeration

There are a total of 198 s-t paths.

There are	6 nodes in path number	1
	1 6 9 11 39 41	
There are	6 nodes in path number	2
	1 6 9 11 38 41	
There are	6 nodes in path number	3
	1 6 9 11 24 41	
There are	6 nodes in path number	4
	1 6 9 11 23 41	
There are	8 nodes in path number	5
	1 6 9 11 12 22 37 41	
There are	8 nodes in path number	6
	1 6 9 11 12 21 36 41	
There are	8 nodes in path number	7
	1 6 9 11 12 20 35 41	
There are	8 nodes in path number	8
	1 6 9 11 12 19 39 41	
There are	8 nodes in path number	9
	1 6 9 11 12 18 38 41	
There are	8 nodes in path number	10
	1 6 9 11 12 17 34 41	
There are	8 nodes in path number	11
	1 6 9 11 12 17 33 41	
There are	8 nodes in path number	12
	1 6 9 11 12 17 30 41	
There are	8 nodes in path number	13
	1 6 9 11 12 16 36 41	
There are	8 nodes in path number	14
	1 6 9 11 12 16 33 41	
There are	8 nodes in path number	15
	1 6 9 11 12 16 32 41	
There are	8 nodes in path number	16
	1 6 9 11 12 16 31 41	
There are	8 nodes in path number	17
	1 6 9 11 12 16 30 41	
There are	8 nodes in path number	18
	1 6 9 11 12 15 32 41	

There are 8 nodes in path number 19  
1 6 9 11 12 15 31 41

There are 8 nodes in path number 20  
1 6 9 11 12 14 29 41

There are 8 nodes in path number 21  
1 6 9 11 12 14 28 41

There are 8 nodes in path number 22  
1 6 9 11 12 14 27 41

There are 8 nodes in path number 23  
1 6 9 11 12 14 26 41

There are 8 nodes in path number 24  
1 6 9 11 12 14 25 41

There are 8 nodes in path number 25  
1 6 9 11 12 14 24 41

There are 8 nodes in path number 26  
1 6 9 11 12 14 23 41

There are 8 nodes in path number 27  
1 6 9 11 12 13 29 41

There are 8 nodes in path number 28  
1 6 9 11 12 13 28 41

There are 8 nodes in path number 29  
1 6 9 11 12 13 27 41

There are 8 nodes in path number 30  
1 6 9 11 12 13 26 41

There are 8 nodes in path number 31  
1 6 9 11 12 13 25 41

There are 8 nodes in path number 32  
1 6 9 11 12 13 24 41

There are 8 nodes in path number 33  
1 6 9 11 12 13 23 41

There are 6 nodes in path number 34  
1 5 8 11 39 41

There are 6 nodes in path number 35  
1 5 8 11 38 41

There are 6 nodes in path number 36  
1 5 8 11 24 41

There are 6 nodes in path number 37  
1 5 8 11 23 41

There are 8 nodes in path number 38  
1 5 8 11 12 22 37 41

There are 8 nodes in path number 39  
1 5 8 11 12 21 36 41

There are 8 nodes in path number 40  
1 5 8 11 12 20 35 41

There are 8 nodes in path number 41  
1 5 8 11 12 19 39 41

There are 8 nodes in path number 42  
1 5 8 11 12 18 38 41

There are 8 nodes in path number 43  
1 5 8 11 12 17 34 41

There are 8 nodes in path number 44  
1 5 8 11 12 17 33 41

There are 8 nodes in path number 45  
1 5 8 11 12 17 30 41

There are 8 nodes in path number 46  
1 5 8 11 12 16 36 41

There are 8 nodes in path number 47  
1 5 8 11 12 16 33 41

There are 8 nodes in path number 48  
1 5 8 11 12 16 32 41

There are 8 nodes in path number 49  
1 5 8 11 12 16 31 41

There are 8 nodes in path number 50  
1 5 8 11 12 16 30 41

There are 8 nodes in path number 51  
1 5 8 11 12 15 32 41

There are 8 nodes in path number 52  
1 5 8 11 12 15 31 41

There are 8 nodes in path number 53  
1 5 8 11 12 14 29 41

There are 8 nodes in path number 54  
1 5 8 11 12 14 28 41

There are 8 nodes in path number 55  
1 5 8 11 12 14 27 41

There are 8 nodes in path number 56  
1 5 8 11 12 14 26 41

There are 8 nodes in path number 57  
1 5 8 11 12 14 25 41

There are 8 nodes in path number 58  
1 5 8 11 12 14 24 41

There are 8 nodes in path number 59  
           1 5 8 11 12 14 23 41  
  
 There are 8 nodes in path number 60  
           1 5 8 11 12 13 29 41  
  
 There are 8 nodes in path number 61  
           1 5 8 11 12 13 28 41  
  
 There are 8 nodes in path number 62  
           1 5 8 11 12 13 27 41  
  
 There are 8 nodes in path number 63  
           1 5 8 11 12 13 26 41  
  
 There are 8 nodes in path number 64  
           1 5 8 11 12 13 25 41  
  
 There are 8 nodes in path number 65  
           1 5 8 11 12 13 24 41  
  
 There are 8 nodes in path number 66  
           1 5 8 11 12 13 23 41  
  
 There are 5 nodes in path number 67  
           1 4 11 39 41  
  
 There are 5 nodes in path number 68  
           1 4 11 38 41  
  
 There are 5 nodes in path number 69  
           1 4 11 24 41  
  
 There are 5 nodes in path number 70  
           1 4 11 23 41  
  
 There are 7 nodes in path number 71  
           1 4 11 12 22 37 41  
  
 There are 7 nodes in path number 72  
           1 4 11 12 21 36 41  
  
 There are 7 nodes in path number 73  
           1 4 11 12 20 35 41  
  
 There are 7 nodes in path number 74  
           1 4 11 12 19 39 41  
  
 There are 7 nodes in path number 75  
           1 4 11 12 18 38 41  
  
 There are 7 nodes in path number 76  
           1 4 11 12 17 34 41  
  
 There are 7 nodes in path number 77  
           1 4 11 12 17 33 41  
  
 There are 7 nodes in path number 78  
           1 4 11 12 17 30 41

There are 7 nodes in path number 79  
           1 4 11 12 16 36 41  
  
 There are 7 nodes in path number 80  
           1 4 11 12 16 33 41  
  
 There are 7 nodes in path number 81  
           1 4 11 12 16 32 41  
  
 There are 7 nodes in path number 82  
           1 4 11 12 16 31 41  
  
 There are 7 nodes in path number 83  
           1 4 11 12 16 30 41  
  
 There are 7 nodes in path number 84  
           1 4 11 12 15 32 41  
  
 There are 7 nodes in path number 85  
           1 4 11 12 15 31 41  
  
 There are 7 nodes in path number 86  
           1 4 11 12 14 29 41  
  
 There are 7 nodes in path number 87  
           1 4 11 12 14 28 41  
  
 There are 7 nodes in path number 88  
           1 4 11 12 14 27 41  
  
 There are 7 nodes in path number 89  
           1 4 11 12 14 26 41  
  
 There are 7 nodes in path number 90  
           1 4 11 12 14 25 41  
  
 There are 7 nodes in path number 91  
           1 4 11 12 14 24 41  
  
 There are 7 nodes in path number 92  
           1 4 11 12 14 23 41  
  
 There are 7 nodes in path number 93  
           1 4 11 12 13 29 41  
  
 There are 7 nodes in path number 94  
           1 4 11 12 13 28 41  
  
 There are 7 nodes in path number 95  
           1 4 11 12 13 27 41  
  
 There are 7 nodes in path number 96  
           1 4 11 12 13 26 41  
  
 There are 7 nodes in path number 97  
           1 4 11 12 13 25 41  
  
 There are 7 nodes in path number 98  
           1 4 11 12 13 24 41

There are 7 nodes in path number 99  
1 4 11 12 13 23 41

There are 7 nodes in path number 100  
1 3 7 10 11 39 41

There are 7 nodes in path number 101  
1 3 7 10 11 38 41

There are 7 nodes in path number 102  
1 3 7 10 11 24 41

There are 7 nodes in path number 103  
1 3 7 10 11 23 41

There are 9 nodes in path number 104  
1 3 7 10 11 12 22 37 41

There are 9 nodes in path number 105  
1 3 7 10 11 12 21 36 41

There are 9 nodes in path number 106  
1 3 7 10 11 12 20 35 41

There are 9 nodes in path number 107  
1 3 7 10 11 12 19 39 41

There are 9 nodes in path number 108  
1 3 7 10 11 12 18 38 41

There are 9 nodes in path number 109  
1 3 7 10 11 12 17 34 41

There are 9 nodes in path number 110  
1 3 7 10 11 12 17 33 41

There are 9 nodes in path number 111  
1 3 7 10 11 12 17 30 41

There are 9 nodes in path number 112  
1 3 7 10 11 12 16 36 41

There are 9 nodes in path number 113  
1 3 7 10 11 12 16 33 41

There are 9 nodes in path number 114  
1 3 7 10 11 12 16 32 41

There are 9 nodes in path number 115  
1 3 7 10 11 12 16 31 41

There are 9 nodes in path number 116  
1 3 7 10 11 12 16 30 41

There are 9 nodes in path number 117  
1 3 7 10 11 12 15 32 41

There are 9 nodes in path number 118  
1 3 7 10 11 12 15 31 41

There are 9 nodes in path number 119  
1 3 7 10 11 12 14 29 41

There are 9 nodes in path number 120  
1 3 7 10 11 12 14 28 41

There are 9 nodes in path number 121  
1 3 7 10 11 12 14 27 41

There are 9 nodes in path number 122  
1 3 7 10 11 12 14 26 41

There are 9 nodes in path number 123  
1 3 7 10 11 12 14 25 41

There are 9 nodes in path number 124  
1 3 7 10 11 12 14 24 41

There are 9 nodes in path number 125  
1 3 7 10 11 12 14 23 41

There are 9 nodes in path number 126  
1 3 7 10 11 12 13 29 41

There are 9 nodes in path number 127  
1 3 7 10 11 12 13 28 41

There are 9 nodes in path number 128  
1 3 7 10 11 12 13 27 41

There are 9 nodes in path number 129  
1 3 7 10 11 12 13 26 41

There are 9 nodes in path number 130  
1 3 7 10 11 12 13 25 41

There are 9 nodes in path number 131  
1 3 7 10 11 12 13 24 41

There are 9 nodes in path number 132  
1 3 7 10 11 12 13 23 41

There are 5 nodes in path number 133  
1 2 11 39 41

There are 5 nodes in path number 134  
1 2 11 38 41

There are 5 nodes in path number 135  
1 2 11 24 41

There are 5 nodes in path number 136  
1 2 11 23 41

There are 7 nodes in path number 137  
1 2 11 12 22 37 41

There are 7 nodes in path number 138  
1 2 11 12 21 36 41

There are 7 nodes in path number 139  
1 2 11 12 20 35 41

There are 7 nodes in path number 140  
1 2 11 12 19 39 41

There are 7 nodes in path number 141  
1 2 11 12 18 38 41

There are 7 nodes in path number 142  
1 2 11 12 17 34 41

There are 7 nodes in path number 143  
1 2 11 12 17 33 41

There are 7 nodes in path number 144  
1 2 11 12 17 30 41

There are 7 nodes in path number 145  
1 2 11 12 16 36 41

There are 7 nodes in path number 146  
1 2 11 12 16 33 41

There are 7 nodes in path number 147  
1 2 11 12 16 32 41

There are 7 nodes in path number 148  
1 2 11 12 16 31 41

There are 7 nodes in path number 149  
1 2 11 12 16 30 41

There are 7 nodes in path number 150  
1 2 11 12 15 32 41

There are 7 nodes in path number 151  
1 2 11 12 15 31 41

There are 7 nodes in path number 152  
1 2 11 12 14 29 41

There are 7 nodes in path number 153  
1 2 11 12 14 28 41

There are 7 nodes in path number 154  
1 2 11 12 14 27 41

There are 7 nodes in path number 155  
1 2 11 12 14 26 41

There are 7 nodes in path number 156  
1 2 11 12 14 25 41

There are 7 nodes in path number 157  
1 2 11 12 14 24 41

There are 7 nodes in path number 158  
1 2 11 12 14 23 41



There are 7 nodes in path number 159  
1 2 11 12 13 29 41

There are 7 nodes in path number 160  
1 2 11 12 13 28 41

There are 7 nodes in path number 161  
1 2 11 12 13 27 41

There are 7 nodes in path number 162  
1 2 11 12 13 26 41

There are 7 nodes in path number 163  
1 2 11 12 13 25 41

There are 7 nodes in path number 164  
1 2 11 12 13 24 41

There are 7 nodes in path number 165  
1 2 11 12 13 23 41

There are 5 nodes in path number 166  
1 40 11 39 41

There are 5 nodes in path number 167  
1 40 11 38 41

There are 5 nodes in path number 168  
1 40 11 24 41

There are 5 nodes in path number 169  
1 40 11 23 41

There are 7 nodes in path number 170  
1 40 11 12 22 37 41

There are 7 nodes in path number 171  
1 40 11 12 21 36 41

There are 7 nodes in path number 172  
1 40 11 12 20 35 41

There are 7 nodes in path number 173  
1 40 11 12 19 39 41

There are 7 nodes in path number 174  
1 40 11 12 18 38 41

There are 7 nodes in path number 175  
1 40 11 12 17 34 41

There are 7 nodes in path number 176  
1 40 11 12 17 33 41

There are 7 nodes in path number 177  
1 40 11 12 17 30 41

There are 7 nodes in path number 178  
1 40 11 12 16 36 41

There are 7 nodes in path number 179  
1 40 11 12 16 33 41

There are 7 nodes in path number 180  
1 40 11 12 16 32 41

There are 7 nodes in path number 181  
1 40 11 12 16 31 41

There are 7 nodes in path number 182  
1 40 11 12 16 30 41

There are 7 nodes in path number 183  
1 40 11 12 15 32 41

There are 7 nodes in path number 184  
1 40 11 12 15 31 41

There are 7 nodes in path number 185  
1 40 11 12 14 29 41

There are 7 nodes in path number 186  
1 40 11 12 14 28 41

There are 7 nodes in path number 187  
1 40 11 12 14 27 41

There are 7 nodes in path number 188  
1 40 11 12 14 26 41

There are 7 nodes in path number 189  
1 40 11 12 14 25 41

There are 7 nodes in path number 190  
1 40 11 12 14 24 41

There are 7 nodes in path number 191  
1 40 11 12 14 23 41

There are 7 nodes in path number 192  
1 40 11 12 13 29 41

There are 7 nodes in path number 193  
1 40 11 12 13 28 41

There are 7 nodes in path number 194  
1 40 11 12 13 27 41

There are 7 nodes in path number 195  
1 40 11 12 13 26 41

There are 7 nodes in path number 196  
1 40 11 12 13 25 41

There are 7 nodes in path number 197  
1 40 11 12 13 24 41

There are 7 nodes in path number 198  
1 40 11 12 13 23 41

### E.3 Program Output For Network C

OPTIONS:

- 1) Obtain dynamic performance bounds
- 2) Investigate steady state performance bounds
- 3) Estimate vector time-series performance data
- 4) Exit this program

Enter 1, 2, 3, or 4 to choose an option.

OPTIONS:

- 1) Obtain dynamic performance bounds
- 2) Investigate steady state performance bounds
- 3) Estimate vector time-series performance data
- 4) Exit this program

Enter 1, 2, 3, or 4 to choose an option.

1

#### GUIDELINES FOR SETTING PERIOD AND UTILIZATION LEVEL

All of the dynamic models will require a period to be set.

This period should be greater than the longest path.

The length of the longest path is 6

The relative performance models call for a utilization level.

This is a constant value for the level of s-t flow.

To avoid problems, this value should be kept well under the  
the lower bound on the max expected flow.

Using an arbitrarily large period, 600, the lower bound on  
expected per unit time flow is 5190.92

Utilization level should be set with respect to period.

This value should be period times some level of flow less than  
the stated lower bound on expected max flow.

The following would be a valid parameter selection:

PERIOD = 12

UTILIZATION = 12 \* 2595 = 31140

Now set parameters according to the guidelines.

Enter a value for the dynamic period.

(must be greater than 6)

10

Enter a value for the utilization level.

(must be less than 51909.2)

25000

Below are the bounds on expected network performance

when period is set to 10

and per time period flow is 2500.00

Reliability	Throughput	Bit Error Rate	Average Delay
0.840830 1.000000	38994.66 43453.49	0.094364 0.119064	2.24954 2.29679

Do you want to compute bounds for other parameters?

(Enter y or n)

y

Enter a value for the dynamic period.

(must be greater than 6)

10

Enter a value for the utilization level.

(must be less than 51909.2)

12500

Below are the bounds on expected network performance

when period is set to 10

and per time period flow is 1250.00

Reliability	Throughput	Bit Error Rate	Average Delay
0.840830 1.000000	38994.66 43453.49	0.037837 0.100696	2.00000 2.00000

Do you want to compute bounds for other parameters?

(Enter y or n)

y

Enter a value for the dynamic period.

(must be greater than 6)

10

Enter a value for the utilization level.

(must be less than 51909.2)

37500

Below are the bounds on expected network performance

when period is set to 10

and per time period flow is 3750.00

Reliability	Throughput	Bit Error Rate	Average Delay
0.840830 1.000000	38994.66 43453.49	0.149639 0.168344	3.06138 3.14074

Do you want to compute bounds for other parameters?

(Enter y or n)

y

Enter a value for the dynamic period.

(must be greater than 6)

10

Enter a value for the utilization level.

(must be less than 51909.2)

50000

Network failure

Network failure

Below are the bounds on expected network performance

when period is set to 10

and per time period flow is 5000.00

Reliability	Throughput	Bit Error Rate	Average Delay
0.840830 1.000000	38994.66 43453.49	0.155960 0.191326	3.23950 3.35933

Do you want to compute bounds for other parameters?

(Enter y or n)

n

OPTIONS:

- 1) Obtain dynamic performance bounds
- 2) Investigate steady state performance bounds
- 3) Estimate vector time-series performance data
- 4) Exit this program

Enter 1, 2, 3, or 4 to choose an option.

4

## ***Appendix F. Validation Data and Output***

### ***F.1 Original Estimated Data Set***

Rel	BER
-----	-----
0.955749	0.257420
0.952337	0.254780
0.990716	0.255701
0.992331	0.259367
0.993679	0.265508
0.988184	0.261902
0.988184	0.261902
0.968715	0.261902
0.972180	0.266593
0.993650	0.272615
0.993650	0.272615
0.993650	0.272615
0.990553	0.260581
0.996944	0.266593
0.995015	0.260581
0.990553	0.260581
0.993315	0.269079
0.989476	0.269079
0.979716	0.269079
0.979716	0.269079
0.993815	0.272615
0.995796	0.275061
0.998857	0.275061
0.997890	0.272615
0.993315	0.269079
1.000000	0.252112
0.998312	0.271548
0.998312	0.271548
0.993679	0.265508
0.978474	0.261902
0.972514	0.261902
0.972514	0.261902
0.968816	0.268002
0.931487	0.263943
0.950096	0.261902
0.914058	0.263943
0.885495	0.266230
0.911382	0.264018
0.919183	0.268280
0.921625	0.268640
0.950214	0.266242
0.964724	0.268235
0.977709	0.265398
0.990809	0.260581
0.990809	0.260581
0.976257	0.272615
0.975358	0.275061
0.967785	0.275061
0.983108	0.277490
0.968542	0.274001

0.956053 0.274001  
0.962012 0.274001  
0.964338 0.270478  
0.965580 0.268002  
0.964338 0.270352  
0.939766 0.268326  
0.934719 0.263943  
0.961798 0.266043  
0.980536 0.261902  
0.968389 0.265508  
0.961390 0.269079  
0.939766 0.268326  
0.966363 0.268002  
0.951353 0.264420  
0.974205 0.270478  
0.979963 0.270478  
0.990080 0.270478  
0.987001 0.270478  
0.990080 0.270478  
0.976880 0.268002  
0.933124 0.266043  
0.938274 0.269079  
0.959097 0.269079  
0.959097 0.269079  
0.950087 0.269079  
0.968431 0.260581  
0.961946 0.260581  
0.986120 0.260581  
0.987134 0.269079  
0.985174 0.271548  
0.947375 0.268002  
0.946420 0.265508  
0.938719 0.261902  
0.916631 0.261902  
0.935765 0.265508  
0.973504 0.257039  
0.973504 0.257039  
0.932109 0.261787  
0.956893 0.264018  
0.941672 0.264018  
0.993019 0.249852  
0.995143 0.249852  
0.986400 0.249852  
0.995121 0.249852  
0.987776 0.253463  
0.993315 0.269079  
0.994240 0.272615  
0.991876 0.260581  
0.977709 0.265398  
0.990809 0.260581  
0.990809 0.260581  
0.986193 0.265398  
0.971793 0.268235  
0.982068 0.270153  
0.968515 0.269079  
0.968515 0.269079  
0.991278 0.271548  
0.987134 0.269079  
0.975753 0.260581  
0.988064 0.266593

0.993061 0.264090  
0.975753 0.264090  
0.989614 0.264090  
0.989614 0.264090  
0.989614 0.264090  
0.993934 0.266254  
0.989614 0.264090  
0.962620 0.260581  
0.946230 0.266593  
0.959095 0.269079  
0.915441 0.266043  
0.899806 0.266043  
0.887505 0.266043  
0.896744 0.268326  
0.930560 0.268002  
0.933039 0.271548  
0.938274 0.269079  
0.968515 0.269079  
0.991876 0.272615  
0.990702 0.275061  
0.972209 0.271548  
0.968715 0.268002  
0.948901 0.268002  
0.936003 0.268002  
0.916391 0.266043  
0.897676 0.266043  
0.873033 0.268702  
0.888198 0.268702  
0.939300 0.268280  
0.969489 0.273304  
0.967742 0.270631  
0.950787 0.270631  
0.957901 0.270631  
0.937977 0.273304  
0.905482 0.268280  
0.917459 0.266043  
0.916391 0.266043  
0.950331 0.265508  
0.929938 0.265508  
0.929938 0.265508  
0.910755 0.264018  
0.933657 0.260581  
0.950696 0.260581  
0.987447 0.264090  
0.977136 0.264090  
0.958038 0.266242  
0.948768 0.268640  
0.915492 0.270749  
0.882154 0.271329  
0.890469 0.273513  
0.904796 0.275348  
0.919061 0.274525  
0.934981 0.271548  
0.950206 0.271548  
0.983919 0.271548  
0.978309 0.271548  
0.968715 0.268002  
0.992982 0.249852  
0.983394 0.249852  
0.980662 0.249852



0.967830 0.253463  
0.961581 0.258373  
0.939766 0.259632  
0.908921 0.256636  
0.895514 0.259338  
0.916170 0.259338  
0.936642 0.261787  
0.946090 0.264018  
0.960178 0.265398  
0.987562 0.269079  
0.969990 0.269079  
0.964245 0.269079  
0.987776 0.271548  
0.969990 0.269079  
0.960315 0.269079  
0.977925 0.271548  
0.965244 0.271548  
0.990648 0.275061  
0.988378 0.275061  
0.975402 0.271548  
0.975402 0.271548  
0.963511 0.271548  
0.968155 0.274001  
0.985094 0.274001  
0.956310 0.271548  
0.944299 0.272156  
0.963511 0.271548  
0.981336 0.275061  
0.974186 0.275061  
0.990648 0.275061  
0.990648 0.275061  
0.985852 0.275061  
0.992414 0.271548  
0.996777 0.248602  
0.994312 0.248602  
0.988957 0.248488  
0.993713 0.248488  
0.990556 0.248488  
0.990556 0.248488  
0.989285 0.252112  
1.000000 0.255701  
0.975508 0.259826  
0.975508 0.259826  
0.967823 0.257420  
0.954570 0.255392  
0.954570 0.255392  
0.923710 0.256636  
0.931554 0.259338  
0.951951 0.261902  
0.968300 0.258373  
0.974157 0.258373  
0.982178 0.263219  
0.969402 0.268002  
0.977925 0.271548  
0.974119 0.268002  
0.991692 0.270478  
0.968816 0.258373  
0.983394 0.253463  
0.987075 0.253463  
1.000000 0.252112

1.000000	0.262998
1.000000	0.260470
1.000000	0.264090
1.000000	0.264090
1.000000	0.259256
1.000000	0.260581
0.996944	0.260581
0.996944	0.260581
0.989614	0.260581
0.991876	0.272615
0.994240	0.275061
0.990702	0.275061
0.987906	0.275061
0.995796	0.275061
0.978472	0.271548
0.969990	0.269079
0.964245	0.269079
0.940106	0.266043
0.929018	0.266217
0.929018	0.266217
0.936697	0.266217
0.931167	0.268768
0.935230	0.270875
0.930949	0.270875
0.928743	0.268326
0.940821	0.270250
0.970900	0.271548
0.981150	0.271548
0.986170	0.269079
1.000000	0.252112
1.000000	0.252112
1.000000	0.252112
1.000000	0.252112
1.000000	0.248488
1.000000	0.275061
0.998556	0.252112
0.998556	0.252112
0.999102	0.275061
0.996705	0.275061
0.990809	0.272615

## ***F.2 Observed Data and Aggregate Estimated Data***

<u>Reliability</u>		<u>Bit Error Rate</u>	
OBS	EST	OBS	EST
0.9697	0.9882	0.2817	0.2658
0.9757	0.9563	0.2390	0.2676
0.9732	0.9603	0.2584	0.2661
0.9746	0.9835	0.3292	0.2633
0.9691	0.9342	0.3604	0.2687
0.9733	0.9457	0.3139	0.2642
0.9640	0.9788	0.3276	0.2670
0.9714	0.9796	0.3476	0.2608
0.9788	0.9735	0.3081	0.2665

### F.3 Output From Estimating Equation Calibration

STUDENT EDITION OF STATISTIX 4.0  
02/05/95, 16:55

ESTIMATE,

UNWEIGHTED LEAST SQUARES LINEAR REGRESSION OF R

NOTE: MODEL FORCED THROUGH ORIGIN

PREDICTOR VARIABLES	COEFFICIENT	STD ERROR	STUDENT'S T	P	VIF
BFIT	2.67541	1.15035	2.33	0.0590	0.1
R1	0.26261	0.31617	0.83	0.4380	0.6

R-SQUARED	0.9996	RESID. MEAN SQUARE (MSE)	4.583E-04
ADJUSTED R-SQUARED	0.9995	STANDARD DEVIATION	0.02141

SOURCE	DF	SS	MS	F	P
REGRESSION	2	7.43365	3.71683	8109.54	0.0000
RESIDUAL	6	0.00275	4.583E-04		
TOTAL	8	7.43640			

CASES INCLUDED 8 MISSING CASES 1

STUDENT EDITION OF STATISTIX 4.0  
02/05/95, 16:57

ESTIMATE,

UNWEIGHTED LEAST SQUARES LINEAR REGRESSION OF REL

NOTE: MODEL FORCED THROUGH ORIGIN

PREDICTOR VARIABLES	COEFFICIENT	STD ERROR	STUDENT'S T	P	VIF
BERFIT	0.03543	0.05680	0.62	0.5557	1.0
REL1	0.98993	0.01816	54.52	0.0000	0.0

R-SQUARED	1.0000	RESID. MEAN SQUARE (MSE)	4.352E-05
ADJUSTED R-SQUARED	1.0000	STANDARD DEVIATION	0.00660

SOURCE	DF	SS	MS	F	P
REGRESSION	2	7.56612	3.78306	86933.26	0.0000
RESIDUAL	6	2.611E-04	4.352E-05		
TOTAL	8	7.56638			

CASES INCLUDED 8 MISSING CASES 1

STUDENT EDITION OF STATISTIX 4.0  
02/05/95, 16:58

ESTIMATE,

UNWEIGHTED LEAST SQUARES LINEAR REGRESSION OF B

NOTE: MODEL FORCED THROUGH ORIGIN

PREDICTOR VARIABLES	COEFFICIENT	STD ERROR	STUDENT'S T	P	VIF
RFIT	0.11044	0.05598	1.97	0.0960	0.6
B1	0.59933	0.20325	2.95	0.0257	0.1

R-SQUARED	0.9998	RESID. MEAN SQUARE (MSE)	1.430E-05
ADJUSTED R-SQUARED	0.9998	STANDARD DEVIATION	0.00378

SOURCE	DF	SS	MS	F	P
REGRESSION	2	0.56399	0.28199	19716.24	0.0000
RESIDUAL	6	8.582E-05	1.430E-05		
TOTAL	8	0.56408			

CASES INCLUDED 8 MISSING CASES 1

STUDENT EDITION OF STATISTIX 4.0  
02/05/95, 17:03

ESTIMATE,

UNWEIGHTED LEAST SQUARES LINEAR REGRESSION OF BER

NOTE: MODEL FORCED THROUGH ORIGIN

PREDICTOR VARIABLES	COEFFICIENT	STD ERROR	STUDENT'S T	P	VIF
RELFIT	0.16068	0.10693	1.50	0.1836	0.0
BER1	0.50216	0.33557	1.50	0.1852	1.0

R-SQUARED	0.9884	RESID. MEAN SQUARE (MSE)	0.00151
ADJUSTED R-SQUARED	0.9846	STANDARD DEVIATION	0.03889

SOURCE	DF	SS	MS	F	P
REGRESSION	2	0.77468	0.38734	256.05	0.0000
RESIDUAL	6	0.00908	0.00151		
TOTAL	8	0.78376			

CASES INCLUDED 8 MISSING CASES 1

#### F.4 Output From Time-Series Residual Analysis

STUDENT EDITION OF STATISTIX 4.0  
02/05/95, 22:31

ESTIMATE,

AUTOCORRELATION PLOT FOR EREL

LAG	CORR.	-1.0	-0.8	-0.6	-0.4	-0.2	0.0	0.2	0.4	0.6	0.8	1.0
1	-0.093						***					
2	-0.406						*****					
3	0.318						*****					
4	-0.272						*****					

MEAN OF THE SERIES	0.00552
STD. DEV. OF SERIES	0.01821
NUMBER OF CASES	9



STUDENT EDITION OF STATISTIX 4.0  
02/05/95, 22:32

ESTIMATE,

AUTOCORRELATION PLOT FOR EBER

LAG	CORR.	-1.0	-0.8	-0.6	-0.4	-0.2	0.0	0.2	0.4	0.6	0.8	1.0
1	0.493											
2	-0.146											
3	-0.269											
4	-0.264											

MEAN OF THE SERIES 0.04177  
STD. DEV. OF SERIES 0.03886  
NUMBER OF CASES 9

STUDENT EDITION OF STATISTIX 4.0  
02/05/95, 22:32

ESTIMATE,

PARTIAL AUTOCORRELATION PLOT FOR EREL

LAG	CORR.	-1.0	-0.8	-0.6	-0.4	-0.2	0.0	0.2	0.4	0.6	0.8	1.0
		+-----+	+-----+	+-----+	+-----+	+-----+	+-----+	+-----+	+-----+	+-----+	+-----+	+-----+
1	-0.093				>		***					<
2	-0.418				>	*****						<
3	0.277				>		*****					<
4	-0.525				*<*****							<

MEAN OF THE SERIES      0.00552  
STD. DEV. OF SERIES    0.01821  
NUMBER OF CASES        9

STUDENT EDITION OF STATISTIX 4.0  
02/05/95, 22:32

ESTIMATE,

PARTIAL AUTOCORRELATION PLOT FOR EBER

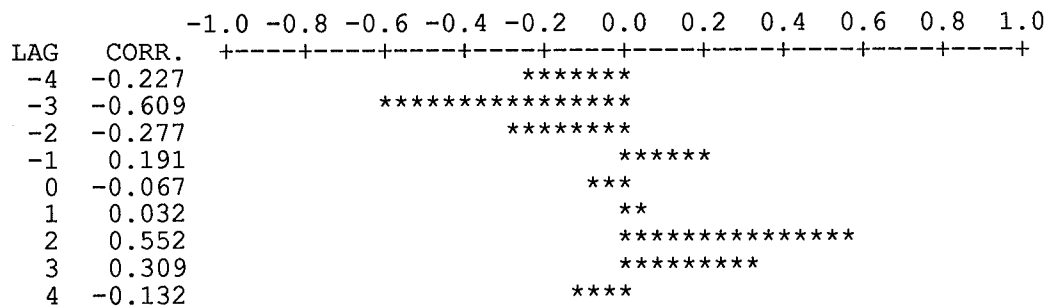
LAG	CORR.	-1.0	-0.8	-0.6	-0.4	-0.2	0.0	0.2	0.4	0.6	0.8	1.0
1	0.493											
2	-0.514											
3	0.167											
4	-0.398											

MEAN OF THE SERIES 0.04177  
STD. DEV. OF SERIES 0.03886  
NUMBER OF CASES 9

STUDENT EDITION OF STATISTIX 4.0  
02/05/95, 22:33

ESTIMATE,

CROSS CORRELATION PLOT FOR EREL AND EBER



MEAN OF THE SERIES 1 0.00552  
STD. DEV. OF SERIES 1 0.01821  
MEAN OF THE SERIES 2 0.04177  
STD. DEV. OF SERIES 2 0.03886  
NUMBER OF CASES 9

STUDENT EDITION OF STATISTIX 4.0  
02/06/95, 9:44

ESTIMATE,

UNWEIGHTED LEAST SQUARES LINEAR REGRESSION OF EB

PREDICTOR VARIABLES	COEFFICIENT	STD ERROR	STUDENT'S T	P
CONSTANT	0.02440	0.02007	1.22	0.2697
EBLAG	0.49289	0.34197	1.44	0.1996
R-SQUARED	0.2572	RESID. MEAN SQUARE (MSE)		0.00159
ADJUSTED R-SQUARED	0.1334	STANDARD DEVIATION		0.03986

SOURCE	DF	SS	MS	F	P
REGRESSION	1	0.00330	0.00330	2.08	0.1996
RESIDUAL	6	0.00953	0.00159		
TOTAL	7	0.01284			

CASES INCLUDED 8 MISSING CASES 1

## *Vita*

Captain John Van Hove was born on 14 May 1967 in Berlin, New Jersey. He graduated from Buena Regional High School in Buena, New Jersey in 1985 and went on to attend the U. S. Air Force Academy, graduating with a Bachelor of Science in Operations Research in June 1989. Upon graduation, he received a regular commission in the USAF and served his first tour of duty with Rome Laboratory at Griffiss AFB, New York. He began as a Neural Systems Analyst, performing research in the field of neural networks with a team serving as the lab's artificial neural systems focal point. He went on to work as the lab's Advanced Planning System Lead Engineer where he led the development, testing, and fielding of a multi-million dollar force level air combat planning tool. In August 1993, he entered the School of Engineering, Air Force Institute of Technology.

Permanent Address: 5521 Honeyleaf Way  
Riverside, OH 45424

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188	
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.				
1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE March 1995	3. REPORT TYPE AND DATES COVERED Master's Thesis		
4. TITLE AND SUBTITLE PROACTIVE MONITORING OF PERFORMANCE IN STOCHASTIC COMMUNICATION NETWORKS		5. FUNDING NUMBERS		
6. AUTHOR(S) John C. Van Hove, Captain, USAF				
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)  Air Force Institute of Technology, WPAFB OH 45433-6583		8. PERFORMING ORGANIZATION REPORT NUMBER  AFIT/GOR/ENS/95M-18		
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)  Lt Col Larry Pulcher R&E Bldg, R5 9800 Savage Rd Ft Mead MD 20755		10. SPONSORING / MONITORING AGENCY REPORT NUMBER		
11. SUPPLEMENTARY NOTES				
12a. DISTRIBUTION / AVAILABILITY STATEMENT  Approved for public release; distribution unlimited		12b. DISTRIBUTION CODE		
13. ABSTRACT (Maximum 200 words)  This research proposes several models for placing bounds on the expected values of some dynamic performance measures for computer communication networks with failing components. These models provide an understanding of expected network performance that is useful in the process of proactive performance monitoring and also in defining level of service agreements with network users. There were three objectives for this research. The first objective was to extend some existing models of steady-state stochastic network performance to a dynamic network flow representation in order to capture the dynamic nature of proactive monitoring. The second objective was to convert the extended absolute performance models to relative performance models that are dependent on the utilization level of the network. This was accomplished by converting a maximum network flow model for throughput to a minimum cost flow model with a constant level of source to sink network flow. The final objective was to demonstrate a methodology for validating the proposed models against an operational communication network. This was accomplished by collecting actual vector time-series performance data, using the models to estimate a similar data set, and performing some multivariate analysis with the two data sets.				
14. SUBJECT TERMS  Stochastic Networks, Network Performance, Dynamic Performance, Performance Monitoring, Network Reliability		15. NUMBER OF PAGES 190		
		16. PRICE CODE		
17. SECURITY CLASSIFICATION OF REPORT  Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE  Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT  Unclassified	20. LIMITATION OF ABSTRACT  UL	

## GENERAL INSTRUCTIONS FOR COMPLETING SF 298

The Report Documentation Page (RDP) is used in announcing and cataloging reports. It is important that this information be consistent with the rest of the report, particularly the cover and title page. Instructions for filling in each block of the form follow. It is important to *stay within the lines* to meet optical scanning requirements.

**Block 1. Agency Use Only (Leave blank).**

**Block 2. Report Date.** Full publication date including day, month, and year, if available (e.g. 1 Jan 88). Must cite at least the year.

**Block 3. Type of Report and Dates Covered.** State whether report is interim, final, etc. If applicable, enter inclusive report dates (e.g. 10 Jun 87 - 30 Jun 88).

**Block 4. Title and Subtitle.** A title is taken from the part of the report that provides the most meaningful and complete information. When a report is prepared in more than one volume, repeat the primary title, add volume number, and include subtitle for the specific volume. On classified documents enter the title classification in parentheses.

**Block 5. Funding Numbers.** To include contract and grant numbers; may include program element number(s), project number(s), task number(s), and work unit number(s). Use the following labels:

C - Contract	PR - Project
G - Grant	TA - Task
PE - Program Element	WU - Work Unit Accession No.

**Block 6. Author(s).** Name(s) of person(s) responsible for writing the report, performing the research, or credited with the content of the report. If editor or compiler, this should follow the name(s).

**Block 7. Performing Organization Name(s) and Address(es).** Self-explanatory.

**Block 8. Performing Organization Report Number.** Enter the unique alphanumeric report number(s) assigned by the organization performing the report.

**Block 9. Sponsoring/Monitoring Agency Name(s) and Address(es).** Self-explanatory.

**Block 10. Sponsoring/Monitoring Agency Report Number.** (If known)

**Block 11. Supplementary Notes.** Enter information not included elsewhere such as: Prepared in cooperation with...; Trans. of...; To be published in.... When a report is revised, include a statement whether the new report supersedes or supplements the older report.

**Block 12a. Distribution/Availability Statement.** Denotes public availability or limitations. Cite any availability to the public. Enter additional limitations or special markings in all capitals (e.g. NOFORN, REL, ITAR).

**DOD** - See DoDD 5230.24, "Distribution Statements on Technical Documents."

**DOE** - See authorities.

**NASA** - See Handbook NHB 2200.2.

**NTIS** - Leave blank.

**Block 12b. Distribution Code.**

**DOD** - Leave blank.

**DOE** - Enter DOE distribution categories from the Standard Distribution for Unclassified Scientific and Technical Reports.

**NASA** - Leave blank.

**NTIS** - Leave blank.

**Block 13. Abstract.** Include a brief (*Maximum 200 words*) factual summary of the most significant information contained in the report.

**Block 14. Subject Terms.** Keywords or phrases identifying major subjects in the report.

**Block 15. Number of Pages.** Enter the total number of pages.

**Block 16. Price Code.** Enter appropriate price code (*NTIS only*).

**Blocks 17. - 19. Security Classifications.** Self-explanatory. Enter U.S. Security Classification in accordance with U.S. Security Regulations (i.e., UNCLASSIFIED). If form contains classified information, stamp classification on the top and bottom of the page.

**Block 20. Limitation of Abstract.** This block must be completed to assign a limitation to the abstract. Enter either UL (unlimited) or SAR (same as report). An entry in this block is necessary if the abstract is to be limited. If blank, the abstract is assumed to be unlimited.